Particular ATC

A02 Drugs for acid related disorders

```
        for param_seasonal in seasonal_pdq:
            try:
                mod = sm.tsa.statespace.SARIMAX(y,
                                                order=param,
                                                seasonal_order=param_seasonal,
                                                enforce_stationarity=False,
                                                enforce_invertibility=False)

                results = mod.fit()

                print('ARIMA{}x{}12 - AIC:{}'.format(param, param_seasonal, results.aic))
            except:
                continue
```

```
ARIMA(0, 0, 0)x(0, 0, 0, 12)12 - AIC:890.1893698252876
ARIMA(0, 0, 0)x(0, 0, 1, 12)12 - AIC:565.4150495709108
ARIMA(0, 0, 0)x(0, 1, 0, 12)12 - AIC:488.61552023538275
ARIMA(0, 0, 0)x(0, 1, 1, 12)12 - AIC:226.91563704043097
ARIMA(0, 0, 0)x(1, 0, 0, 12)12 - AIC:506.58439992497085
ARIMA(0, 0, 0)x(1, 0, 1, 12)12 - AIC:487.45073518958975
ARIMA(0, 0, 0)x(1, 1, 0, 12)12 - AIC:244.91781409333925
ARIMA(0, 0, 0)x(1, 1, 1, 12)12 - AIC:227.4309002911438
ARIMA(0, 0, 1)x(0, 0, 0, 12)12 - AIC:841.7531652617959
ARIMA(0, 0, 1)x(0, 0, 1, 12)12 - AIC:544.8059393202336
ARIMA(0, 0, 1)x(0, 1, 0, 12)12 - AIC:466.1501613756076
ARIMA(0, 0, 1)x(0, 1, 1, 12)12 - AIC:208.54303111143912
ARIMA(0, 0, 1)x(1, 0, 0, 12)12 - AIC:591.0522114625408
ARIMA(0, 0, 1)x(1, 0, 1, 12)12 - AIC:544.2489265865637
ARIMA(0, 0, 1)x(1, 1, 0, 12)12 - AIC:247.82250916942868
ARIMA(0, 0, 1)x(1, 1, 1, 12)12 - AIC:206.23406439629193
ARIMA(0, 1, 0)x(0, 0, 0, 12)12 - AIC:715.5535027767473
ARIMA(0, 1, 0)x(0, 0, 1, 12)12 - AIC:450.6934871778215
ARIMA(0, 1, 0)x(0, 1, 0, 12)12 - AIC:466.64894655619077
ARIMA(0, 1, 0)x(0, 1, 1, 12)12 - AIC:209.5591478088914
ARIMA(0, 1, 0)x(1, 0, 0, 12)12 - AIC:464.18008807307257
ARIMA(0, 1, 0)x(1, 0, 1, 12)12 - AIC:452.32260504922726
ARIMA(0, 1, 0)x(1, 1, 0, 12)12 - AIC:228.62534785418762
ARIMA(0, 1, 0)x(1, 1, 1, 12)12 - AIC:210.86393485813971
ARIMA(0, 1, 1)x(0, 0, 0, 12)12 - AIC:695.046720268938
ARIMA(0, 1, 1)x(0, 0, 1, 12)12 - AIC:422.07021234262965
ARIMA(0, 1, 1)x(0, 1, 0, 12)12 - AIC:436.34996578505263
ARIMA(0, 1, 1)x(0, 1, 1, 12)12 - AIC:186.03117024629802
ARIMA(0, 1, 1)x(1, 0, 0, 12)12 - AIC:463.3246571002089
ARIMA(0, 1, 1)x(1, 0, 1, 12)12 - AIC:422.70083961659304
ARIMA(0, 1, 1)x(1, 1, 0, 12)12 - AIC:226.08209352365705
ARIMA(0, 1, 1)x(1, 1, 1, 12)12 - AIC:187.9893516183192
ARIMA(1, 0, 0)x(0, 0, 0, 12)12 - AIC:737.7769302128178
ARIMA(1, 0, 0)x(0, 0, 1, 12)12 - AIC:473.02317541914545
ARIMA(1, 0, 0)x(0, 1, 0, 12)12 - AIC:482.6830819161955
ARIMA(1, 0, 0)x(0, 1, 1, 12)12 - AIC:229.47228814400626
ARIMA(1, 0, 0)x(1, 0, 0, 12)12 - AIC:474.0272287464602
ARIMA(1, 0, 0)x(1, 0, 1, 12)12 - AIC:474.7747122747352
ARIMA(1, 0, 0)x(1, 1, 0, 12)12 - AIC:228.22177370530966
ARIMA(1, 0, 0)x(1, 1, 1, 12)12 - AIC:229.48731494588048
ARIMA(1, 0, 1)x(0, 0, 0, 12)12 - AIC:727.0772397435351
ARIMA(1, 0, 1)x(0, 0, 1, 12)12 - AIC:443.13535438619806
ARIMA(1, 0, 1)x(0, 1, 0, 12)12 - AIC:464.1666298725106
ARIMA(1, 0, 1)x(0, 1, 1, 12)12 - AIC:206.0205645972613
ARIMA(1, 0, 1)x(1, 0, 0, 12)12 - AIC:466.3082497166157
ARIMA(1, 0, 1)x(1, 0, 1, 12)12 - AIC:444.9078450877433
ARIMA(1, 0, 1)x(1, 1, 0, 12)12 - AIC:227.35806741441394
ARIMA(1, 0, 1)x(1, 1, 1, 12)12 - AIC:207.57566652733314
ARIMA(1, 1, 0)x(0, 0, 0, 12)12 - AIC:716.779357006552
ARIMA(1, 1, 0)x(0, 0, 1, 12)12 - AIC:452.9650091339171
ARIMA(1, 1, 0)x(0, 1, 0, 12)12 - AIC:467.80699409582263
ARIMA(1, 1, 0)x(0, 1, 1, 12)12 - AIC:210.60037868282473
ARIMA(1, 1, 0)x(1, 0, 0, 12)12 - AIC:453.8014088435554
ARIMA(1, 1, 0)x(1, 0, 1, 12)12 - AIC:454.6733451016035
ARIMA(1, 1, 0)x(1, 1, 0, 12)12 - AIC:210.90829922949393
ARIMA(1, 1, 0)x(1, 1, 1, 12)12 - AIC:212.18960693488302
ARIMA(1, 1, 1)x(0, 0, 0, 12)12 - AIC:706.7035572614831
ARIMA(1, 1, 1)x(0, 0, 1, 12)12 - AIC:423.29342274244544
ARIMA(1, 1, 1)x(0, 1, 0, 12)12 - AIC:433.39131412280165
ARIMA(1, 1, 1)x(0, 1, 1, 12)12 - AIC:187.7711601683244
ARIMA(1, 1, 1)x(1, 0, 0, 12)12 - AIC:444.13324370360453
ARIMA(1, 1, 1)x(1, 0, 1, 12)12 - AIC:424.5557390740716
ARIMA(1, 1, 1)x(1, 1, 0, 12)12 - AIC:207.6832936972972
ARIMA(1, 1, 1)x(1, 1, 1, 12)12 - AIC:189.4249761734752
```

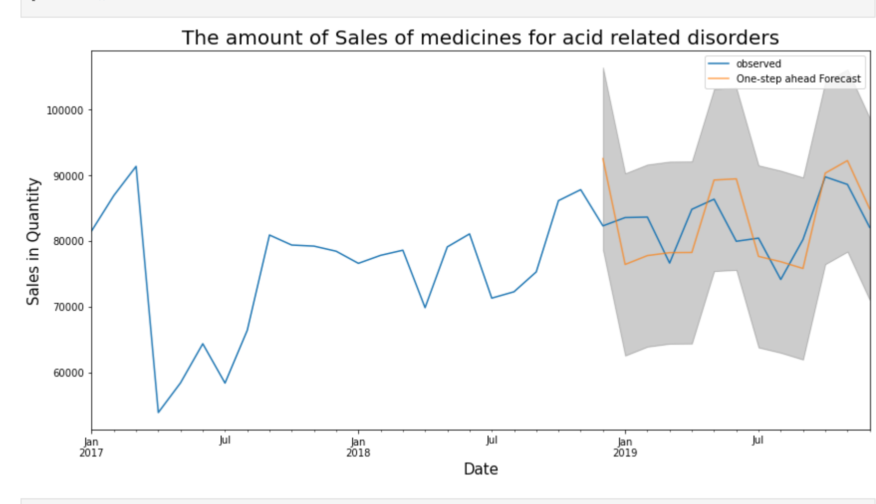
In [275…

```
mod = sm.tsa.statespace.SARIMAX(y,
                                order=(0, 1, 1),
                                seasonal_order=(0, 1, 1, 12),
                                enforce_stationarity=False,
                                enforce_invertibility=False)

results = mod.fit()

print(results.summary().tables[1])
```

	coef	std err	z	P>|z|	[0.025	0.975]
ma.L1	-0.8505	0.167	-5.087	0.000	-1.178	-0.523
ma.S.L12	0.2420	0.324	0.746	0.456	-0.394	0.878
sigma2	4.99e+07	1.59e-10	3.14e+17	0.000	4.99e+07	4.99e+07

In [277…

```
pred = results.get_prediction(start=pd.to_datetime('2018-12-01'), dynamic=False)
pred_ci = pred.conf_int()
ax = y['2013':].plot(label='observed')
pred.predicted_mean.plot(ax=ax, label='One-step ahead Forecast', alpha=.7, figsize=(14, 7))

ax.fill_between(pred_ci.index,
 pred_ci.iloc[:, 0],
 pred_ci.iloc[:, 1], color='k', alpha=.2)
ax.set_xlabel('Date',fontsize=15)
ax.set_ylabel('Sales in Quantity',fontsize=15)
ax.set_title ('The amount of Sales of medicines for acid related disorders',fontsize=20)
plt.legend()
plt.show()
```



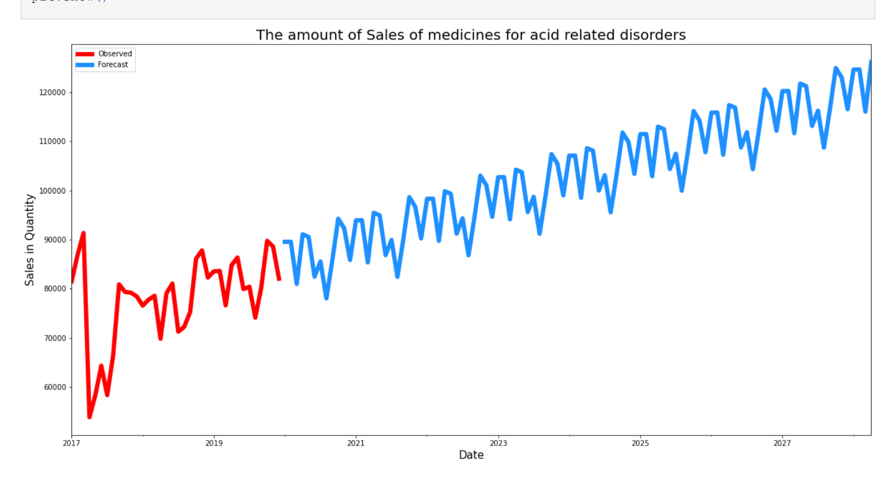
In [278…

```
y_forecasted = pred.predicted_mean
y_truth = y['2018-12-01':]
# Compute the mean squared error
mse = ((y_forecasted - y_truth) ** 2).mean()
print('The Mean Squared Error of our forecasts is {}'.format(round(mse, 2)))
print('The Root Mean Squared Error of our forecasts is {}'.format(round(np.sqrt(mse), 2)))
```

```
The Mean Squared Error of our forecasts is 29982136.72
The Root Mean Squared Error of our forecasts is 5475.59
```

In [279…

```
pred_uc = results.get_forecast(steps=100)
pred_ci = pred_uc.conf_int()
ax = y.plot(label='Observed', figsize=(20, 10),color='r',linewidth=6)
pred_uc.predicted_mean.plot(ax=ax, label='Forecast',color='dodgerblue',linewidth=6)
ax.set_xlabel('Date',fontsize=15)
ax.set_ylabel('Sales in Quantity',fontsize=15)
ax.set_title ('The amount of Sales of medicines for acid related disorders',fontsize=20)
plt.legend()
plt.show()
```



B01 Antithrombotic agents

In [299…

```
df =df.loc[(df["ATCShort"]=="B01")]
```

In [300…

```
df = df.sort_values('Delivery date')
```

In [301…

```
df = df.groupby('Delivery date')['Units'].sum().reset_index()
```

In [302…

```
df = df.set_index('Delivery date')
```

In [303…

```
y = df['Units'].resample('MS').mean()
```

In [304…

```
import itertools
import statsmodels.api as sm
p = d = q = range(0, 2)

# Generate all different combinations of p, q and q triplets
pdq = list(itertools.product(p, d, q))

# Generate all different combinations of seasonal p, q and q triplets
seasonal_pdq = [(x[0], x[1], x[2], 12) for x in list(itertools.product(p, d, q))]

print('Examples of parameter combinations for Seasonal ARIMA...')
print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[1]))
print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[2]))
print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[3]))
print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[4]))
```

```
Examples of parameter combinations for Seasonal ARIMA...
SARIMAX: (0, 0, 1) x (0, 0, 1, 12)
SARIMAX: (0, 0, 1) x (0, 1, 0, 12)
SARIMAX: (0, 1, 0) x (0, 1, 1, 12)
SARIMAX: (0, 1, 0) x (1, 0, 0, 12)
```

In [305…

```
for param in pdq:
    for param_seasonal in seasonal_pdq:
        try:
            mod = sm.tsa.statespace.SARIMAX(y,
                                            order=param,
                                            seasonal_order=param_seasonal,
                                            enforce_stationarity=False,
                                            enforce_invertibility=False)

            results = mod.fit()

            print('ARIMA{}x{}12 - AIC:{}'.format(param, param_seasonal, results.aic))
        except:
            continue
```

```
ARIMA(0, 0, 0)x(0, 0, 0, 12)12 - AIC:896.0673609781238
ARIMA(0, 0, 0)x(0, 0, 1, 12)12 - AIC:568.6654466537661
ARIMA(0, 0, 0)x(0, 1, 0, 12)12 - AIC:478.3529179086874
ARIMA(0, 0, 0)x(0, 1, 1, 12)12 - AIC:226.24136958207527
ARIMA(0, 0, 0)x(1, 0, 0, 12)12 - AIC:498.2693887526009
ARIMA(0, 0, 0)x(1, 0, 1, 12)12 - AIC:479.04756132240084
ARIMA(0, 0, 0)x(1, 1, 0, 12)12 - AIC:245.8690452705066
ARIMA(0, 0, 0)x(1, 1, 1, 12)12 - AIC:226.4879552057049
ARIMA(0, 0, 1)x(0, 0, 0, 12)12 - AIC:848.0400346971867
ARIMA(0, 0, 1)x(0, 0, 1, 12)12 - AIC:548.4034664776593
ARIMA(0, 0, 1)x(0, 1, 0, 12)12 - AIC:457.7757570223302
ARIMA(0, 0, 1)x(0, 1, 1, 12)12 - AIC:208.5102130248407
ARIMA(0, 0, 1)x(1, 0, 0, 12)12 - AIC:634.7711509810224
ARIMA(0, 0, 1)x(1, 0, 1, 12)12 - AIC:547.8280583968041
ARIMA(0, 0, 1)x(1, 1, 0, 12)12 - AIC:247.43192741220003
ARIMA(0, 0, 1)x(1, 1, 1, 12)12 - AIC:206.56347696394124
ARIMA(0, 1, 0)x(0, 0, 0, 12)12 - AIC:709.5442319264256
ARIMA(0, 1, 0)x(0, 0, 1, 12)12 - AIC:454.49972952020835
ARIMA(0, 1, 0)x(0, 1, 0, 12)12 - AIC:465.62396026115755
ARIMA(0, 1, 0)x(0, 1, 1, 12)12 - AIC:213.8111995420354
ARIMA(0, 1, 0)x(1, 0, 0, 12)12 - AIC:474.7354094057255
ARIMA(0, 1, 0)x(1, 0, 1, 12)12 - AIC:456.03838547024515
ARIMA(0, 1, 0)x(1, 1, 0, 12)12 - AIC:234.30005242599958
ARIMA(0, 1, 0)x(1, 1, 1, 12)12 - AIC:214.01700140053637
ARIMA(0, 1, 1)x(0, 0, 0, 12)12 - AIC:686.3209738238764
ARIMA(0, 1, 1)x(0, 0, 1, 12)12 - AIC:430.5355982991507
ARIMA(0, 1, 1)x(0, 1, 0, 12)12 - AIC:435.5994441320641
ARIMA(0, 1, 1)x(0, 1, 1, 12)12 - AIC:188.41671152161072
ARIMA(0, 1, 1)x(1, 0, 0, 12)12 - AIC:471.2676780018432
ARIMA(0, 1, 1)x(1, 0, 1, 12)12 - AIC:430.4224939204161
ARIMA(0, 1, 1)x(1, 1, 0, 12)12 - AIC:228.70057981327236
ARIMA(0, 1, 1)x(1, 1, 1, 12)12 - AIC:190.3689878798538
ARIMA(1, 0, 0)x(0, 0, 0, 12)12 - AIC:731.7769369698151
ARIMA(1, 0, 0)x(0, 0, 1, 12)12 - AIC:476.5042671635142
ARIMA(1, 0, 0)x(0, 1, 0, 12)12 - AIC:477.8034993625623
ARIMA(1, 0, 0)x(0, 1, 1, 12)12 - AIC:227.57070855674107
ARIMA(1, 0, 0)x(1, 0, 0, 12)12 - AIC:477.837409579573
ARIMA(1, 0, 0)x(1, 0, 1, 12)12 - AIC:478.16495985346234
ARIMA(1, 0, 0)x(1, 1, 0, 12)12 - AIC:226.50579311846178
ARIMA(1, 0, 0)x(1, 1, 1, 12)12 - AIC:228.0539840261546
ARIMA(1, 0, 1)x(0, 0, 0, 12)12 - AIC:707.4052606846315
ARIMA(1, 0, 1)x(0, 0, 1, 12)12 - AIC:449.82215510371134
ARIMA(1, 0, 1)x(0, 1, 0, 12)12 - AIC:459.491487195816
ARIMA(1, 0, 1)x(0, 1, 1, 12)12 - AIC:210.42314935968972
ARIMA(1, 0, 1)x(1, 0, 0, 12)12 - AIC:473.24404000887415
ARIMA(1, 0, 1)x(1, 0, 1, 12)12 - AIC:451.7042012051508
ARIMA(1, 0, 1)x(1, 1, 0, 12)12 - AIC:227.5722840197307
ARIMA(1, 0, 1)x(1, 1, 1, 12)12 - AIC:208.43870580008758
ARIMA(1, 1, 0)x(0, 0, 0, 12)12 - AIC:710.8231961401093
ARIMA(1, 1, 0)x(0, 0, 1, 12)12 - AIC:455.6892442938954
ARIMA(1, 1, 0)x(0, 1, 0, 12)12 - AIC:465.08971494963373
ARIMA(1, 1, 0)x(0, 1, 1, 12)12 - AIC:211.66134633284753
ARIMA(1, 1, 0)x(1, 0, 0, 12)12 - AIC:456.6550184740049
ARIMA(1, 1, 0)x(1, 0, 1, 12)12 - AIC:457.60948782803126
ARIMA(1, 1, 0)x(1, 1, 0, 12)12 - AIC:211.91953967277385
ARIMA(1, 1, 0)x(1, 1, 1, 12)12 - AIC:205.58818018035376
ARIMA(1, 1, 1)x(0, 0, 0, 12)12 - AIC:687.7014702973195
ARIMA(1, 1, 1)x(0, 0, 1, 12)12 - AIC:430.85220007050026
ARIMA(1, 1, 1)x(0, 1, 0, 12)12 - AIC:437.51195840380274
ARIMA(1, 1, 1)x(0, 1, 1, 12)12 - AIC:189.271549598417
ARIMA(1, 1, 1)x(1, 0, 0, 12)12 - AIC:452.6138692467982
ARIMA(1, 1, 1)x(1, 0, 1, 12)12 - AIC:432.30093236386176
ARIMA(1, 1, 1)x(1, 1, 0, 12)12 - AIC:209.4832369755357
ARIMA(1, 1, 1)x(1, 1, 1, 12)12 - AIC:183.04272450065957
```

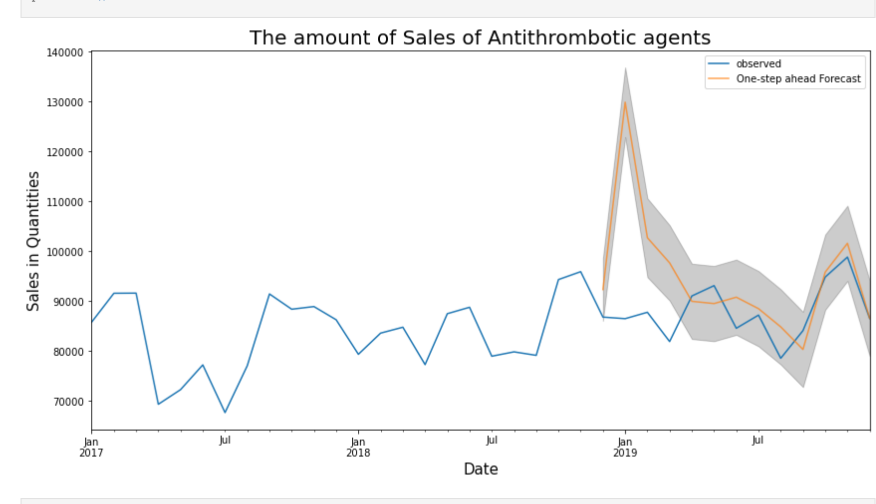
In [306…

```
import statsmodels.api as sm
mod = sm.tsa.statespace.SARIMAX(y,
                                order=(1,1,1),
                                seasonal_order=(1, 1, 1, 12),
                                enforce_stationarity=False,
                                enforce_invertibility=False)

results = mod.fit()

print(results.summary().tables[1])
```

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.6986	0.407	-1.716	0.086	-1.497	0.099
ma.L1	-0.3908	0.191	-2.045	0.041	-0.765	-0.016
ar.S.L12	-1.3252	0.186	-7.131	0.000	-1.689	-0.961
ma.S.L12	1.9238	0.309	6.236	0.000	1.319	2.528
sigma2	6.733e+06	1.91e-08	3.53e+14	0.000	6.73e+06	6.73e+06

In [307…

```
pred = results.get_prediction(start=pd.to_datetime('2018-12-01'), dynamic=False)
pred_ci = pred.conf_int()
ax = y['2013':].plot(label='observed')
pred.predicted_mean.plot(ax=ax, label='One-step ahead Forecast', alpha=.7, figsize=(14, 7))

ax.fill_between(pred_ci.index,
 pred_ci.iloc[:, 0],
 pred_ci.iloc[:, 1], color='k', alpha=.2)
ax.set_xlabel('Date',fontsize=15)
ax.set_ylabel('Sales in Quantities',fontsize=15)
ax.set_title ('The amount of Sales of Antithrombotic agents',fontsize=20)
plt.legend()
plt.show()
```



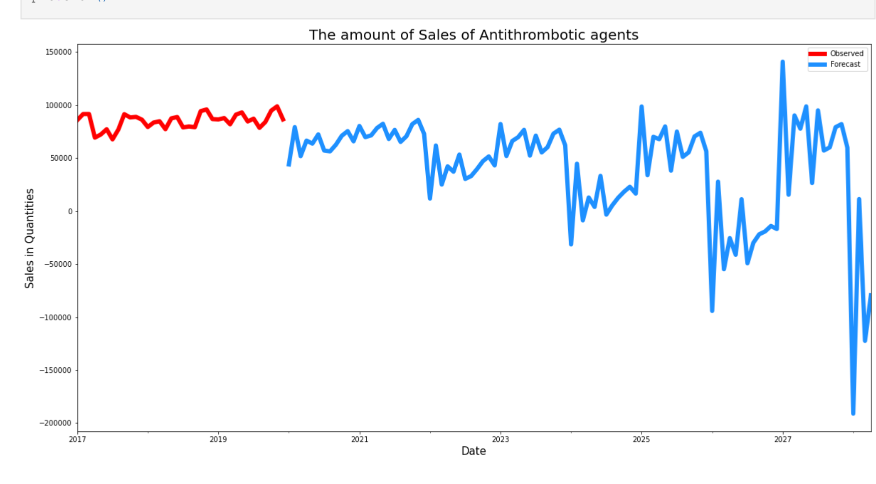
In [308…

```
y_forecasted = pred.predicted_mean
y_truth = y['2018-12-01':]
# Compute the mean squared error
mse = ((y_forecasted - y_truth) ** 2).mean()
print('The Mean Squared Error of our forecasts is {}'.format(round(mse, 2)))
print('The Root Mean Squared Error of our forecasts is {}'.format(round(np.sqrt(mse), 2)))
```

```
The Mean Squared Error of our forecasts is 192690511.41
The Root Mean Squared Error of our forecasts is 13881.3
```

In [309…

```
pred_uc = results.get_forecast(steps=100)
pred_ci = pred_uc.conf_int()
ax = y.plot(label='Observed', figsize=(20, 10),color='r',linewidth=6)
pred_uc.predicted_mean.plot(ax=ax, label='Forecast',color='dodgerblue',linewidth=6)
ax.set_xlabel('Date',fontsize=15)
ax.set_ylabel('Sales in Quantities',fontsize=15)
ax.set_title ('The amount of Sales of Antithrombotic agents',fontsize=20)
plt.legend()
plt.show()
```



C07 Beta blocking agents

In [314…

```
df =df.loc[(df["ATCShort"]=="C07")]
```

In [315…

```
df = df.sort_values('Delivery date')
```

In [316…

```
df = df.groupby('Delivery date')['Units'].sum().reset_index()
```

In [317…

```
df = df.set_index('Delivery date')
```

In [318…

```
y = df['Units'].resample('MS').mean()
```

In [319…

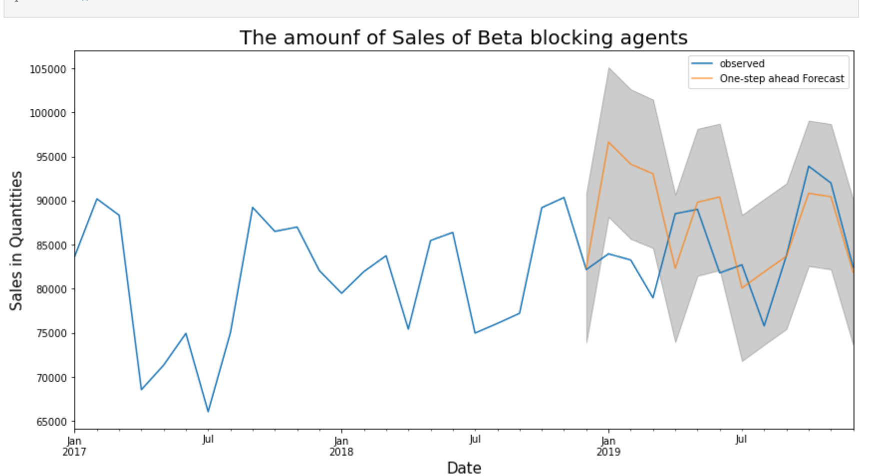
```
import itertools
p = d = q = range(0, 2)

# Generate all different combinations of p, q and q triplets
pdq = list(itertools.product(p, d, q))

# Generate all different combinations of seasonal p, q and q triplets
seasonal_pdq = [(x[0], x[1], x[2], 12) for x in list(itertools.product(p, d, q))]

print('Examples of parameter combinations for Seasonal ARIMA...')
print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[1]))
print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[2]))
print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[3]))
print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[4]))
```

```
Examples of parameter combinations for Seasonal ARIMA...
SARIMAX: (0, 0, 1) x (0, 0, 1, 12)
SARIMAX: (0, 0, 1) x (0, 1, 0, 12)
SARIMAX: (0, 1, 0) x (0, 1, 1, 12)
SARIMAX: (0, 1, 0) x (1, 0, 0, 12)
```

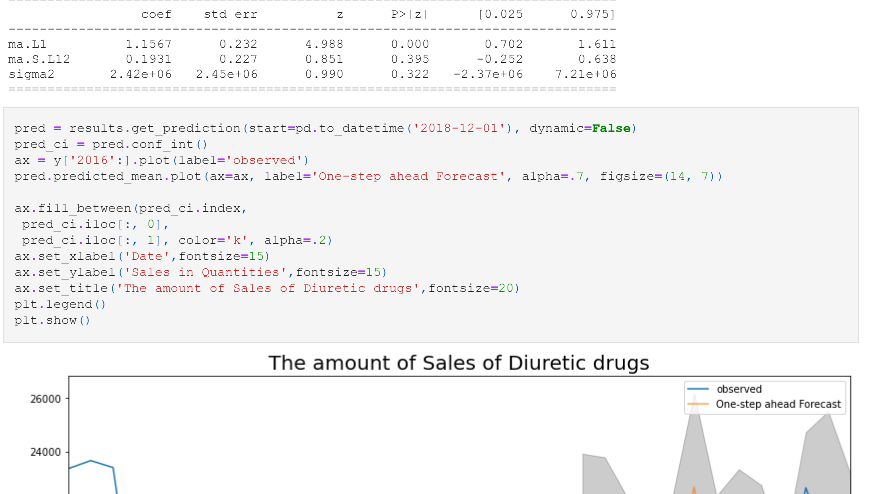
In [320…

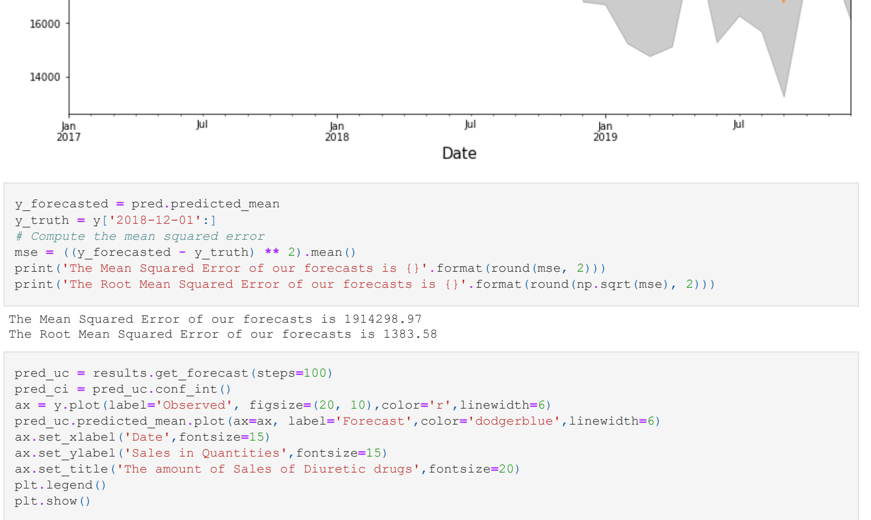
```
import warnings
warnings.filterwarnings("ignore")
for param in pdq:
    for param_seasonal in seasonal_pdq:
        try:
            mod = sm.tsa.statespace.SARIMAX(y,
                                            order=param,
                                            seasonal_order=param_seasonal,
                                            enforce_stationarity=False,
                                            enforce_invertibility=False)

            results = mod.fit()

            print('ARIMA{}x{}12 - AIC:{}'.format(param, param_seasonal, results.aic))
        except:
            continue
```

```
ARIMA(0, 0, 0)x(0, 0, 0, 12)12 - AIC:893.7657990697556
ARIMA(0, 0, 0)x(0, 0, 1, 12)12 - AIC:566.9994057938942
ARIMA(0, 0, 0)x(0, 1, 0, 12)12 - AIC:473.76891702117746
ARIMA(0, 0, 0)x(0, 1, 1, 12)12 - AIC:225.62999289287296
ARIMA(0, 0, 0)x(1, 0, 0, 12)12 - AIC:493.8050736441952
ARIMA(0, 0, 0)x(1, 0, 1, 12)12 - AIC:475.3502768095257
ARIMA(0, 0, 0)x(1, 1, 0, 12)12 - AIC:244.8979015169467
ARIMA(0, 0, 0)x(1, 1, 1, 12)12 - AIC:225.877364984478
ARIMA(0, 0, 1)x(0, 0, 0, 12)12 - AIC:845.7127939761153
ARIMA(0, 0, 1)x(0, 0, 1, 12)12 - AIC:546.9131193179551
ARIMA(0, 0, 1)x(0, 1, 0, 12)12 - AIC:452.8152053316322
ARIMA(0, 0, 1)x(0, 1, 1, 12)12 - AIC:208.26409353966244
ARIMA(0, 0, 1)x(1, 0, 0, 12)12 - AIC:632.860111402267
ARIMA(0, 0, 1)x(1, 0, 1, 12)12 - AIC:546.3098524005709
ARIMA(0, 0, 1)x(1, 1, 0, 12)12 - AIC:246.8556011544573
ARIMA(0, 0, 1)x(1, 1, 1, 12)12 - AIC:207.00613468989897
ARIMA(0, 1, 0)x(0, 0, 0, 12)12 - AIC:703.5120592117548
ARIMA(0, 1, 0)x(0, 0, 1, 12)12 - AIC:450.6457967736724
ARIMA(0, 1, 0)x(0, 1, 0, 12)12 - AIC:461.1928031330283
ARIMA(0, 1, 0)x(0, 1, 1, 12)12 - AIC:214.17303780540846
ARIMA(0, 1, 0)x(1, 0, 0, 12)12 - AIC:470.3859447084332
ARIMA(0, 1, 0)x(1, 0, 1, 12)12 - AIC:452.2409062206534
ARIMA(0, 1, 0)x(1, 1, 0, 12)12 - AIC:234.29254626548172
ARIMA(0, 1, 0)x(1, 1, 1, 12)12 - AIC:213.63781174772086
ARIMA(0, 1, 1)x(0, 0, 0, 12)12 - AIC:680.8079321206502
ARIMA(0, 1, 1)x(0, 0, 1, 12)12 - AIC:426.8931345319276
ARIMA(0, 1, 1)x(0, 1, 0, 12)12 - AIC:433.53384035056484
ARIMA(0, 1, 1)x(0, 1, 1, 12)12 - AIC:186.88079986495342
ARIMA(0, 1, 1)x(1, 0, 0, 12)12 - AIC:467.73011390437256
ARIMA(0, 1, 1)x(1, 0, 1, 12)12 - AIC:428.0375276889837
ARIMA(0, 1, 1)x(1, 1, 0, 12)12 - AIC:228.21758668577272
ARIMA(0, 1, 1)x(1, 1, 1, 12)12 - AIC:184.2939465088077
ARIMA(1, 0, 0)x(0, 0, 0, 12)12 - AIC:725.8367958955585
ARIMA(1, 0, 0)x(0, 0, 1, 12)12 - AIC:472.4266158842965
ARIMA(1, 0, 0)x(0, 1, 0, 12)12 - AIC:473.39068841014046
ARIMA(1, 0, 0)x(0, 1, 1, 12)12 - AIC:227.74140951588006
ARIMA(1, 0, 0)x(1, 0, 0, 12)12 - AIC:473.60405277153205
ARIMA(1, 0, 0)x(1, 0, 1, 12)12 - AIC:474.0884500709542
ARIMA(1, 0, 0)x(1, 1, 0, 12)12 - AIC:227.3698154964428
ARIMA(1, 0, 0)x(1, 1, 1, 12)12 - AIC:226.92488024708035
ARIMA(1, 0, 1)x(0, 0, 0, 12)12 - AIC:702.3459248688612
```



C03 Diuretic drugs





A10 Drugs used in diabetes

```python
y = df['Units'].resample('MS').mean()
```

In [349]:
```python
import itertools
p = d = q = range(0, 2)

# Generate all different combinations of p, q and q triplets
pdq = list(itertools.product(p, d, q))

# Generate all different combinations of seasonal p, q and q triplets
seasonal_pdq = [(x[0], x[1], x[2], 12) for x in list(itertools.product(p, d, q))]

print('Examples of parameter combinations for Seasonal ARIMA...')
print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[1]))
print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[2]))
print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[3]))
print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[4]))
```

```
Examples of parameter combinations for Seasonal ARIMA...
SARIMAX: (0, 0, 1) x (0, 0, 1, 12)
SARIMAX: (0, 0, 1) x (0, 1, 0, 12)
SARIMAX: (0, 1, 0) x (0, 1, 1, 12)
SARIMAX: (0, 1, 0) x (1, 0, 0, 12)
```

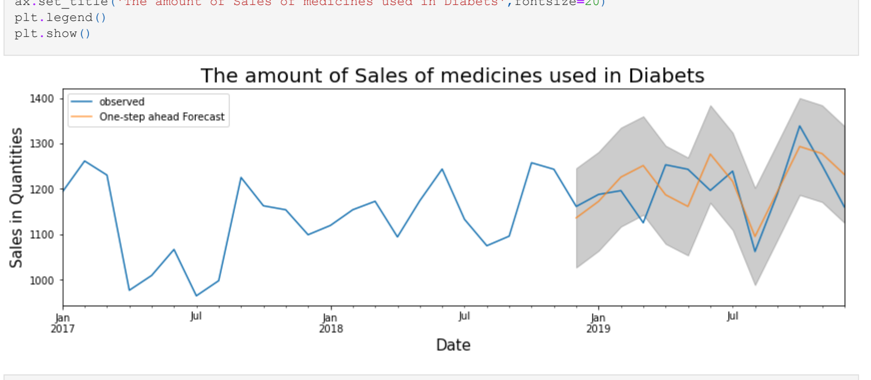
In [350]:
```python
for param in pdq:
    for param_seasonal in seasonal_pdq:
        try:
            mod = sm.tsa.statespace.SARIMAX(y,
                                            order=param,
                                            seasonal_order=param_seasonal,
                                            enforce_stationarity=False,
                                            enforce_invertibility=False)

            results = mod.fit()

            print('ARIMA{}x{}12 - AIC:{}'.format(param, param_seasonal, results.aic))
        except:
            continue
```

```
ARIMA(0, 0, 0)x(0, 0, 0, 12)12 - AIC:595.3166395819993
ARIMA(0, 0, 0)x(0, 0, 1, 12)12 - AIC:375.3642262037889
ARIMA(0, 0, 0)x(0, 1, 0, 12)12 - AIC:278.7045915889768
ARIMA(0, 0, 0)x(0, 1, 1, 12)12 - AIC:129.10159742844596
ARIMA(0, 0, 0)x(1, 0, 0, 12)12 - AIC:287.6586652973614
ARIMA(0, 0, 0)x(1, 0, 1, 12)12 - AIC:276.9969746980637
ARIMA(0, 0, 0)x(1, 1, 0, 12)12 - AIC:140.78081903319753
ARIMA(0, 0, 0)x(1, 1, 1, 12)12 - AIC:129.98818294230415
ARIMA(0, 0, 1)x(0, 0, 0, 12)12 - AIC:537.1083462288468
ARIMA(0, 0, 1)x(0, 0, 1, 12)12 - AIC:335.3420725080202
ARIMA(0, 0, 1)x(0, 1, 0, 12)12 - AIC:265.00605269363604
ARIMA(0, 0, 1)x(0, 1, 1, 12)12 - AIC:119.45662258544519
ARIMA(0, 0, 1)x(1, 0, 0, 12)12 - AIC:287.9056120513402
ARIMA(0, 0, 1)x(1, 0, 1, 12)12 - AIC:264.5679085822349
ARIMA(0, 0, 1)x(1, 1, 0, 12)12 - AIC:142.72012738624628
ARIMA(0, 0, 1)x(1, 1, 1, 12)12 - AIC:120.7599438337709
ARIMA(0, 1, 0)x(0, 0, 0, 12)12 - AIC:409.52674822024994
ARIMA(0, 1, 0)x(0, 0, 1, 12)12 - AIC:263.4416432193273
ARIMA(0, 1, 0)x(0, 1, 0, 12)12 - AIC:271.0561141690568
ARIMA(0, 1, 0)x(0, 1, 1, 12)12 - AIC:126.71349340060858
ARIMA(0, 1, 0)x(1, 0, 0, 12)12 - AIC:272.84222025662604
ARIMA(0, 1, 0)x(1, 0, 1, 12)12 - AIC:263.71804681092124
ARIMA(0, 1, 0)x(1, 1, 0, 12)12 - AIC:137.9840495272948
ARIMA(0, 1, 0)x(1, 1, 1, 12)12 - AIC:128.70679791380942
ARIMA(0, 1, 1)x(0, 0, 0, 12)12 - AIC:393.6943827977473
ARIMA(0, 1, 1)x(0, 0, 1, 12)12 - AIC:242.96561138049708
ARIMA(0, 1, 1)x(0, 1, 0, 12)12 - AIC:255.53676619094645
ARIMA(0, 1, 1)x(0, 1, 1, 12)12 - AIC:106.54716832823716
ARIMA(0, 1, 1)x(1, 0, 0, 12)12 - AIC:264.4592820421401
ARIMA(0, 1, 1)x(1, 0, 1, 12)12 - AIC:242.70914514815072
ARIMA(0, 1, 1)x(1, 1, 0, 12)12 - AIC:129.2931719017987
ARIMA(0, 1, 1)x(1, 1, 1, 12)12 - AIC:108.54577666380314
ARIMA(1, 0, 0)x(0, 0, 0, 12)12 - AIC:422.9173425823452
ARIMA(1, 0, 0)x(0, 0, 1, 12)12 - AIC:276.3974749990728
ARIMA(1, 0, 0)x(0, 1, 0, 12)12 - AIC:276.22564622594325
ARIMA(1, 0, 0)x(0, 1, 1, 12)12 - AIC:131.04133945236077
ARIMA(1, 0, 0)x(1, 0, 0, 12)12 - AIC:274.8276026416589
ARIMA(1, 0, 0)x(1, 0, 1, 12)12 - AIC:276.4473782964738
ARIMA(1, 0, 0)x(1, 1, 0, 12)12 - AIC:131.02360182428973
ARIMA(1, 0, 0)x(1, 1, 1, 12)12 - AIC:131.8804488527305
ARIMA(1, 0, 1)x(0, 0, 0, 12)12 - AIC:403.9220571772916
ARIMA(1, 0, 1)x(0, 0, 1, 12)12 - AIC:253.0586239941553
ARIMA(1, 0, 1)x(0, 1, 0, 12)12 - AIC:266.61699359160207
ARIMA(1, 0, 1)x(0, 1, 1, 12)12 - AIC:121.39493757649406
ARIMA(1, 0, 1)x(1, 0, 0, 12)12 - AIC:264.55058932935736
ARIMA(1, 0, 1)x(1, 0, 1, 12)12 - AIC:253.5952334779744
ARIMA(1, 0, 1)x(1, 1, 0, 12)12 - AIC:132.64490125538083
ARIMA(1, 0, 1)x(1, 1, 1, 12)12 - AIC:122.75980353819351
ARIMA(1, 1, 0)x(0, 0, 0, 12)12 - AIC:410.74493029156
ARIMA(1, 1, 0)x(0, 0, 1, 12)12 - AIC:263.2253722055204
ARIMA(1, 1, 0)x(0, 1, 0, 12)12 - AIC:270.68969569201295
ARIMA(1, 1, 0)x(0, 1, 1, 12)12 - AIC:124.65130013158739
ARIMA(1, 1, 0)x(1, 0, 0, 12)12 - AIC:262.23832064654715
ARIMA(1, 1, 0)x(1, 0, 1, 12)12 - AIC:263.03136784922435
ARIMA(1, 1, 0)x(1, 1, 0, 12)12 - AIC:124.65167166224192
ARIMA(1, 1, 0)x(1, 1, 1, 12)12 - AIC:126.6296918887673
ARIMA(1, 1, 1)x(0, 0, 0, 12)12 - AIC:392.93619587926037
ARIMA(1, 1, 1)x(0, 0, 1, 12)12 - AIC:244.8675274082683
ARIMA(1, 1, 1)x(0, 1, 0, 12)12 - AIC:256.18586986809476
ARIMA(1, 1, 1)x(0, 1, 1, 12)12 - AIC:105.53219262047611
ARIMA(1, 1, 1)x(1, 0, 0, 12)12 - AIC:255.99790853669452
ARIMA(1, 1, 1)x(1, 0, 1, 12)12 - AIC:244.7082461700047
ARIMA(1, 1, 1)x(1, 1, 0, 12)12 - AIC:119.29970899217608
ARIMA(1, 1, 1)x(1, 1, 1, 12)12 - AIC:107.52849254550925
```

In [351]:
```python
import statsmodels.api as sm
mod = sm.tsa.statespace.SARIMAX(y,
                                order=(1,1,1),
                                seasonal_order=(1,1,1,12),
                                enforce_stationarity=False,
                                enforce_invertibility=False)

results = mod.fit()

print(results.summary().tables[1])
```

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.4786	0.311	-1.539	0.124	-1.088	0.131
ma.L1	-1.0000	0.923	-1.084	0.279	-2.809	0.809
ar.S.L12	-0.0009	0.158	-0.006	0.995	-0.310	0.308
ma.S.L12	-0.0907	0.246	-0.369	0.712	-0.573	0.391
sigma2	2797.9656	0.000	8.48e+06	0.000	2797.965	2797.966

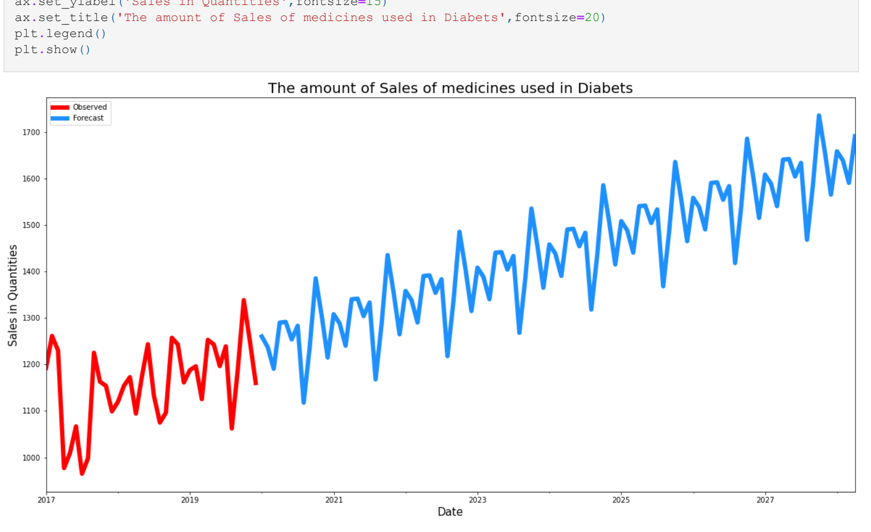
In [352]:
```python
pred = results.get_prediction(start=pd.to_datetime('2018-12-01'), dynamic=False)
pred_ci = pred.conf_int()
ax = y['2016':].plot(label='observed')
pred.predicted_mean.plot(ax=ax, label='One-step ahead Forecast', alpha=.7, figsize=(14, 4))

ax.fill_between(pred_ci.index,
 pred_ci.iloc[:, 0],
 pred_ci.iloc[:, 1], color='k', alpha=.2)
ax.set_xlabel('Date',fontsize=15)
ax.set_ylabel('Sales in Quantities',fontsize=15)
ax.set_title('The amount of Sales of medicines used in Diabets',fontsize=20)
plt.legend()
plt.show()
```



In [353]:
```python
y_forecasted = pred.predicted_mean
y_truth = y['2018-12-01':]
# Compute the mean squared error
mse = ((y_forecasted - y_truth) ** 2).mean()
print('The Mean Squared Error of our forecasts is {}'.format(round(mse, 2)))
print('The Root Mean Squared Error of our forecasts is {}'.format(round(np.sqrt(mse), 2)))
```

```
The Mean Squared Error of our forecasts is 3422.12
The Root Mean Squared Error of our forecasts is 58.5
```

In [355]:
```python
pred_uc = results.get_forecast(steps=100)
pred_ci = pred_uc.conf_int()
ax = y.plot(label='Observed', figsize=(20, 10),color='r',linewidth=6)
pred_uc.predicted_mean.plot(ax=ax, label='Forecast',color='dodgerblue',linewidth=6)
ax.set_xlabel('Date',fontsize=15)
ax.set_ylabel('Sales in Quantities',fontsize=15)
ax.set_title('The amount of Sales of medicines used in Diabets',fontsize=20)
plt.legend()
plt.show()
```



Specific location Forecasting

Brussels

In [359]:
```python
df=df1
```

In [360]:
```python
df =df.loc[(df["Pharmacy Postcode (2)"] >=10) & (df["Pharmacy Postcode (2)"] <=12)]
```

In [361]:
```python
df = df.sort_values('Delivery date')
```

In [362]:
```python
df = df.groupby('Delivery date')['Units'].sum().reset_index()
```

In [363]:
```python
df = df.set_index('Delivery date')
```

In [364]:
```python
y= df['Units'].resample('MS').mean()
```

In [365]:
```python
import itertools
p = d = q = range(0, 2)

# Generate all different combinations of p, q and q triplets
pdq = list(itertools.product(p, d, q))

# Generate all different combinations of seasonal p, q and q triplets
seasonal_pdq = [(x[0], x[1], x[2], 12) for x in list(itertools.product(p, d, q))]

print('Examples of parameter combinations for Seasonal ARIMA...')
print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[1]))
print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[2]))
print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[3]))
print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[4]))
```

```
Examples of parameter combinations for Seasonal ARIMA...
SARIMAX: (0, 0, 1) x (0, 0, 1, 12)
SARIMAX: (0, 0, 1) x (0, 1, 0, 12)
SARIMAX: (0, 1, 0) x (0, 1, 1, 12)
SARIMAX: (0, 1, 0) x (1, 0, 0, 12)
```

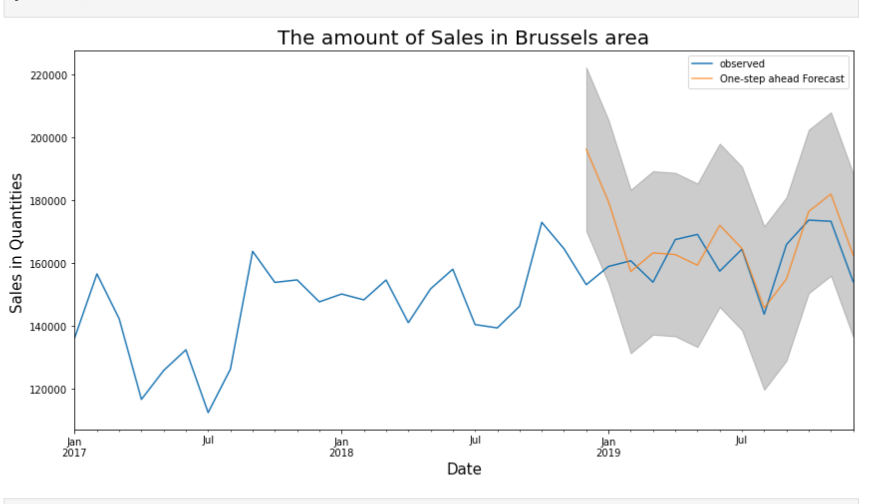
In [366]:
```python
for param in pdq:
    for param_seasonal in seasonal_pdq:
        try:
            mod = sm.tsa.statespace.SARIMAX(y,
                                            order=param,
                                            seasonal_order=param_seasonal,
                                            enforce_stationarity=False,
                                            enforce_invertibility=False)

            results = mod.fit()

            print('ARIMA{}x{}12 - AIC:{}'.format(param, param_seasonal, results.aic))
        except:
            continue
```

```
ARIMA(0, 0, 0)x(0, 0, 0, 12)12 - AIC:936.609269941616
ARIMA(0, 0, 0)x(0, 0, 1, 12)12 - AIC:596.3350103543928
ARIMA(0, 0, 0)x(0, 1, 0, 12)12 - AIC:513.9417984384806
ARIMA(0, 0, 0)x(0, 1, 1, 12)12 - AIC:246.24601732400745
ARIMA(0, 0, 0)x(1, 0, 0, 12)12 - AIC:524.7381282087367
ARIMA(0, 0, 0)x(1, 0, 1, 12)12 - AIC:506.8158855071904
ARIMA(0, 0, 0)x(1, 1, 0, 12)12 - AIC:263.1502353669254
ARIMA(0, 0, 0)x(1, 1, 1, 12)12 - AIC:240.46096898838636
ARIMA(0, 0, 1)x(0, 0, 0, 12)12 - AIC:887.7972151556331
ARIMA(0, 0, 1)x(0, 0, 1, 12)12 - AIC:574.3883877329573
ARIMA(0, 0, 1)x(0, 1, 0, 12)12 - AIC:490.4632266732547
ARIMA(0, 0, 1)x(0, 1, 1, 12)12 - AIC:225.51880767226015
ARIMA(0, 0, 1)x(1, 0, 0, 12)12 - AIC:622.9602293027407
ARIMA(0, 0, 1)x(1, 0, 1, 12)12 - AIC:573.5379569355531
ARIMA(0, 0, 1)x(1, 1, 0, 12)12 - AIC:266.6652884461187
ARIMA(0, 0, 1)x(1, 1, 1, 12)12 - AIC:223.28176639875585
ARIMA(0, 1, 0)x(0, 0, 0, 12)12 - AIC:747.351831975033
ARIMA(0, 1, 0)x(0, 0, 1, 12)12 - AIC:481.0599875782106
ARIMA(0, 1, 0)x(0, 1, 0, 12)12 - AIC:492.7843906309351
ARIMA(0, 1, 0)x(0, 1, 1, 12)12 - AIC:223.14494440494929
ARIMA(0, 1, 0)x(1, 0, 0, 12)12 - AIC:501.1099198967584
ARIMA(0, 1, 0)x(1, 0, 1, 12)12 - AIC:482.88119676198596
ARIMA(0, 1, 0)x(1, 1, 0, 12)12 - AIC:248.2142199427712
ARIMA(0, 1, 0)x(1, 1, 1, 12)12 - AIC:225.12666831201955
ARIMA(0, 1, 1)x(0, 0, 0, 12)12 - AIC:721.945964468431
ARIMA(0, 1, 1)x(0, 0, 1, 12)12 - AIC:453.1340526565759
ARIMA(0, 1, 1)x(0, 1, 0, 12)12 - AIC:460.61171157929147
ARIMA(0, 1, 1)x(0, 1, 1, 12)12 - AIC:199.2442142388339
ARIMA(0, 1, 1)x(1, 0, 0, 12)12 - AIC:494.8964638009092
ARIMA(0, 1, 1)x(1, 0, 1, 12)12 - AIC:453.9969247707715
ARIMA(0, 1, 1)x(1, 1, 0, 12)12 - AIC:242.15644997271372
ARIMA(0, 1, 1)x(1, 1, 1, 12)12 - AIC:201.15005100367802
ARIMA(1, 0, 0)x(0, 0, 0, 12)12 - AIC:772.377872068088
ARIMA(1, 0, 0)x(0, 0, 1, 12)12 - AIC:504.312074709814
ARIMA(1, 0, 0)x(0, 1, 0, 12)12 - AIC:510.1117810548759
ARIMA(1, 0, 0)x(0, 1, 1, 12)12 - AIC:246.38949151470553
ARIMA(1, 0, 0)x(1, 0, 0, 12)12 - AIC:504.27114277552243
ARIMA(1, 0, 0)x(1, 0, 1, 12)12 - AIC:506.0908536127075
ARIMA(1, 0, 0)x(1, 1, 0, 12)12 - AIC:243.94857222156017
ARIMA(1, 0, 0)x(1, 1, 1, 12)12 - AIC:242.80750681727878
ARIMA(1, 0, 1)x(0, 0, 0, 12)12 - AIC:737.6745504610639
ARIMA(1, 0, 1)x(0, 0, 1, 12)12 - AIC:474.00538970410304
ARIMA(1, 0, 1)x(0, 1, 0, 12)12 - AIC:477.5933143453971
ARIMA(1, 0, 1)x(0, 1, 1, 12)12 - AIC:221.10890716485852
ARIMA(1, 0, 1)x(1, 0, 0, 12)12 - AIC:496.75930770945575
ARIMA(1, 0, 1)x(1, 0, 1, 12)12 - AIC:475.6660453929229
ARIMA(1, 0, 1)x(1, 1, 0, 12)12 - AIC:242.23839696645697
ARIMA(1, 0, 1)x(1, 1, 1, 12)12 - AIC:221.4216804464176
ARIMA(1, 1, 0)x(0, 0, 0, 12)12 - AIC:748.0540636912242
ARIMA(1, 1, 0)x(0, 0, 1, 12)12 - AIC:479.44044704826536
ARIMA(1, 1, 0)x(0, 1, 0, 12)12 - AIC:487.36579301069264
ARIMA(1, 1, 0)x(0, 1, 1, 12)12 - AIC:222.10314337553055
ARIMA(1, 1, 0)x(1, 0, 0, 12)12 - AIC:478.77684152127813
ARIMA(1, 1, 0)x(1, 0, 1, 12)12 - AIC:480.77595166450294
ARIMA(1, 1, 0)x(1, 1, 0, 12)12 - AIC:222.25209782003267
ARIMA(1, 1, 0)x(1, 1, 1, 12)12 - AIC:223.2844802627161
ARIMA(1, 1, 1)x(0, 0, 0, 12)12 - AIC:723.8851719443062
ARIMA(1, 1, 1)x(0, 0, 1, 12)12 - AIC:454.7440210587142
ARIMA(1, 1, 1)x(0, 1, 0, 12)12 - AIC:461.1759735218664
ARIMA(1, 1, 1)x(0, 1, 1, 12)12 - AIC:198.89141765447715
ARIMA(1, 1, 1)x(1, 0, 0, 12)12 - AIC:474.0765212004487
ARIMA(1, 1, 1)x(1, 0, 1, 12)12 - AIC:454.89131786406256
ARIMA(1, 1, 1)x(1, 1, 0, 12)12 - AIC:220.4444545739845
ARIMA(1, 1, 1)x(1, 1, 1, 12)12 - AIC:200.65231816793207
```

In [367]:
```python
import statsmodels.api as sm
mod = sm.tsa.statespace.SARIMAX(y,
                                order=(1, 1, 1),
                                seasonal_order=(0, 1, 1, 12),
                                enforce_stationarity=False,
                                enforce_invertibility=False)

results = mod.fit()

print(results.summary().tables[1])
```

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.6163	1.156	-0.533	0.594	-2.881	1.649
ma.L1	-0.8773	0.122	-7.183	0.000	-1.117	-0.638
ma.S.L12	0.1937	0.330	0.587	0.557	-0.453	0.841
sigma2	1.763e+08	2.11e-09	8.37e+16	0.000	1.76e+08	1.76e+08

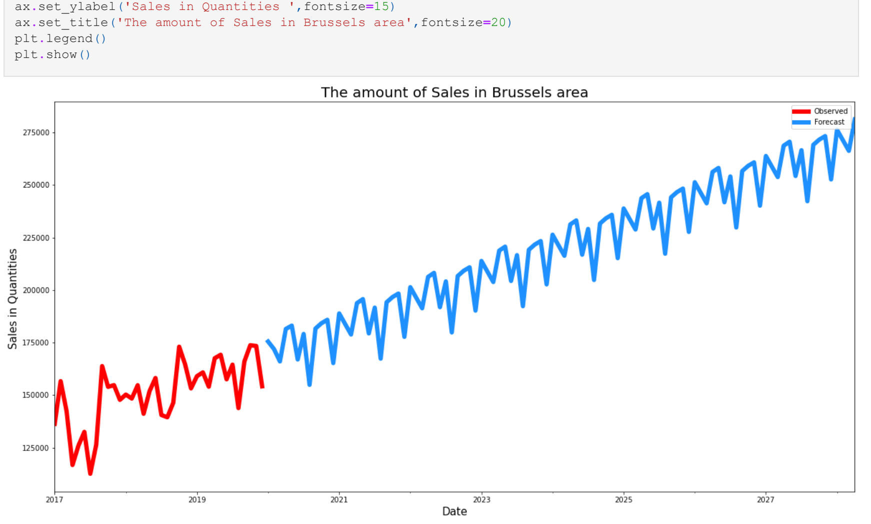
In [368]:
```python
pred = results.get_prediction(start=pd.to_datetime('2018-12-01'), dynamic=False)
pred_ci = pred.conf_int()
ax = y['2013':].plot(label='observed')
pred.predicted_mean.plot(ax=ax, label='One-step ahead Forecast', alpha=.7, figsize=(14, 7))
ax.fill_between(pred_ci.index,
 pred_ci.iloc[:, 0],
 pred_ci.iloc[:, 1], color='k', alpha=.2)
ax.set_xlabel('Date',fontsize=15)
ax.set_ylabel('Sales in Quantities ',fontsize=15)
ax.set_title('The amount of Sales in Brussels area',fontsize=20)
plt.legend()
plt.show()
```



In [369]:
```python
y_forecasted = pred.predicted_mean
y_truth = y['2018-12-01':]
# Compute the mean squared error
mse = ((y_forecasted - y_truth) ** 2).mean()
print('The Mean Squared Error of our forecasts is {}'.format(round(mse, 2)))
print('The Root Mean Squared Error of our forecasts is {}'.format(round(np.sqrt(mse), 2)))
```

```
The Mean Squared Error of our forecasts is 230157023.29
The Root Mean Squared Error of our forecasts is 15170.93
```

In [370]:
```python
pred_uc = results.get_forecast(steps=100)
pred_ci = pred_uc.conf_int()
ax = y.plot(label='Observed', figsize=(20, 10),color='r',linewidth=6)
pred_uc.predicted_mean.plot(ax=ax, label='Forecast',color='dodgerblue',linewidth=6)
ax.set_xlabel('Date',fontsize=15)
ax.set_ylabel('Sales in Quantities ',fontsize=15)
ax.set_title('The amount of Sales in Brussels area',fontsize=20)
plt.legend()
plt.show()
```



Antwerp

In [371]:
```python
df=df1
```

In [372]:
```python
df =df.loc[(df["Pharmacy Postcode (2)"] >=20) & (df["Pharmacy Postcode (2)"] <=29)]
```

In [373]:
```python
df = df.sort_values('Delivery date')
```

In [374]:
```python
df = df.groupby('Delivery date')['Units'].sum().reset_index()
```

In [375]:
```python
df = df.set_index('Delivery date')
```

In [376]:
```python
y= df['Units'].resample('MS').mean()
```

In [377]:
```python
import itertools
p = d = q = range(0, 2)

# Generate all different combinations of p, q and q triplets
pdq = list(itertools.product(p, d, q))

# Generate all different combinations of seasonal p, q and q triplets
seasonal_pdq = [(x[0], x[1], x[2], 12) for x in list(itertools.product(p, d, q))]

print('Examples of parameter combinations for Seasonal ARIMA...')
print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[1]))
print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[2]))
print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[3]))
print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[4]))
```

```
Examples of parameter combinations for Seasonal ARIMA...
SARIMAX: (0, 0, 1) x (0, 0, 1, 12)
SARIMAX: (0, 0, 1) x (0, 1, 0, 12)
SARIMAX: (0, 1, 0) x (0, 1, 1, 12)
SARIMAX: (0, 1, 0) x (1, 0, 0, 12)
```

In [378]:
```python
for param in pdq:
    for param_seasonal in seasonal_pdq:
        try:
            mod = sm.tsa.statespace.SARIMAX(y,
                                            order=param,
                                            seasonal_order=param_seasonal,
                                            enforce_stationarity=False,
                                            enforce_invertibility=False)

            results = mod.fit()

            print('ARIMA{}x{}12 - AIC:{}'.format(param, param_seasonal, results.aic))
        except:
            continue
```

```
ARIMA(0, 0, 0)x(0, 0, 0, 12)12 - AIC:987.9327666777162
ARIMA(0, 0, 0)x(0, 0, 1, 12)12 - AIC:628.9054542433828
ARIMA(0, 0, 0)x(0, 1, 0, 12)12 - AIC:536.6746493538152
ARIMA(0, 0, 0)x(0, 1, 1, 12)12 - AIC:256.2889215834581
ARIMA(0, 0, 0)x(1, 0, 0, 12)12 - AIC:561.9448977522754
ARIMA(0, 0, 0)x(1, 0, 1, 12)12 - AIC:538.5645721469591
ARIMA(0, 0, 0)x(1, 1, 0, 12)12 - AIC:278.5962700937787
ARIMA(0, 0, 0)x(1, 1, 1, 12)12 - AIC:256.68879224712043
ARIMA(0, 0, 1)x(0, 0, 0, 12)12 - AIC:936.9424918910443
```

```
ARIMA(0, 0, 1)x(0, 0, 1, 12)12 - AIC:606.006906875244
ARIMA(0, 0, 1)x(0, 1, 0, 12)12 - AIC:511.4109620821416
ARIMA(0, 0, 1)x(0, 1, 1, 12)12 - AIC:235.6819232160826
ARIMA(0, 0, 1)x(1, 0, 0, 12)12 - AIC:657.7954034981046
ARIMA(0, 0, 1)x(1, 0, 1, 12)12 - AIC:605.4477967140134
ARIMA(0, 0, 1)x(1, 1, 0, 12)12 - AIC:280.6659690140342
ARIMA(0, 0, 1)x(1, 1, 1, 12)12 - AIC:232.55233382606968
ARIMA(0, 1, 0)x(0, 0, 0, 12)12 - AIC:797.6177177815188
ARIMA(0, 1, 0)x(0, 0, 1, 12)12 - AIC:508.7736547059094
ARIMA(0, 1, 0)x(0, 1, 0, 12)12 - AIC:520.3130922403453
ARIMA(0, 1, 0)x(0, 1, 1, 12)12 - AIC:240.44803088424015
ARIMA(0, 1, 0)x(1, 0, 0, 12)12 - AIC:529.9438917272937
ARIMA(0, 1, 0)x(1, 0, 1, 12)12 - AIC:510.5899466193405
ARIMA(0, 1, 0)x(1, 1, 0, 12)12 - AIC:263.67231499457785
ARIMA(0, 1, 0)x(1, 1, 1, 12)12 - AIC:239.62320554302983
ARIMA(0, 1, 1)x(0, 0, 0, 12)12 - AIC:774.8224200702249
ARIMA(0, 1, 1)x(0, 0, 1, 12)12 - AIC:480.37981270170957
ARIMA(0, 1, 1)x(0, 1, 0, 12)12 - AIC:487.8244585883451
ARIMA(0, 1, 1)x(0, 1, 1, 12)12 - AIC:211.67457591287248
ARIMA(0, 1, 1)x(1, 0, 0, 12)12 - AIC:529.5043854538135
ARIMA(0, 1, 1)x(1, 0, 1, 12)12 - AIC:482.37433369957336
ARIMA(0, 1, 1)x(1, 1, 0, 12)12 - AIC:257.95589924166086
ARIMA(0, 1, 1)x(1, 1, 1, 12)12 - AIC:213.47795557879923
ARIMA(1, 0, 0)x(0, 0, 0, 12)12 - AIC:822.3474826826427
ARIMA(1, 0, 0)x(0, 0, 1, 12)12 - AIC:533.3628500265244
ARIMA(1, 0, 0)x(0, 1, 0, 12)12 - AIC:535.1175763195323
ARIMA(1, 0, 0)x(0, 1, 1, 12)12 - AIC:258.0792898401229
ARIMA(1, 0, 0)x(1, 0, 0, 12)12 - AIC:534.5212656269248
ARIMA(1, 0, 0)x(1, 0, 1, 12)12 - AIC:535.2188207652418
ARIMA(1, 0, 0)x(1, 1, 0, 12)12 - AIC:257.91697494765134
ARIMA(1, 0, 0)x(1, 1, 1, 12)12 - AIC:257.5307068144384
ARIMA(1, 0, 1)x(0, 0, 0, 12)12 - AIC:799.3809969833626
ARIMA(1, 0, 1)x(0, 0, 1, 12)12 - AIC:504.6858552822287
ARIMA(1, 0, 1)x(0, 1, 0, 12)12 - AIC:513.037282170578
ARIMA(1, 0, 1)x(0, 1, 1, 12)12 - AIC:237.28176197544235
ARIMA(1, 0, 1)x(1, 0, 0, 12)12 - AIC:533.6310940302528
ARIMA(1, 0, 1)x(1, 0, 1, 12)12 - AIC:506.4423499857549
ARIMA(1, 0, 1)x(1, 1, 0, 12)12 - AIC:258.9797259726727
ARIMA(1, 0, 1)x(1, 1, 1, 12)12 - AIC:233.11745836321595
ARIMA(1, 1, 0)x(0, 0, 0, 12)12 - AIC:799.1185956209415
ARIMA(1, 1, 0)x(0, 0, 1, 12)12 - AIC:510.8562653193226
ARIMA(1, 1, 0)x(0, 1, 0, 12)12 - AIC:520.7153804825289
ARIMA(1, 1, 0)x(0, 1, 1, 12)12 - AIC:239.4368368946162
ARIMA(1, 1, 0)x(1, 0, 0, 12)12 - AIC:511.72313296728237
ARIMA(1, 1, 0)x(1, 0, 1, 12)12 - AIC:512.771685228397
ARIMA(1, 1, 0)x(1, 1, 0, 12)12 - AIC:239.67795425880013
ARIMA(1, 1, 0)x(1, 1, 1, 12)12 - AIC:240.05357741174936
ARIMA(1, 1, 1)x(0, 0, 0, 12)12 - AIC:762.9892453308491
ARIMA(1, 1, 1)x(0, 0, 1, 12)12 - AIC:482.31238252024275
ARIMA(1, 1, 1)x(0, 1, 0, 12)12 - AIC:489.81662555660944
ARIMA(1, 1, 1)x(0, 1, 1, 12)12 - AIC:212.85583963510814
ARIMA(1, 1, 1)x(1, 0, 0, 12)12 - AIC:507.6668530597765
ARIMA(1, 1, 1)x(1, 0, 1, 12)12 - AIC:484.30869061835455
ARIMA(1, 1, 1)x(1, 1, 0, 12)12 - AIC:236.83919649587452
ARIMA(1, 1, 1)x(1, 1, 1, 12)12 - AIC:214.72253856904229
```

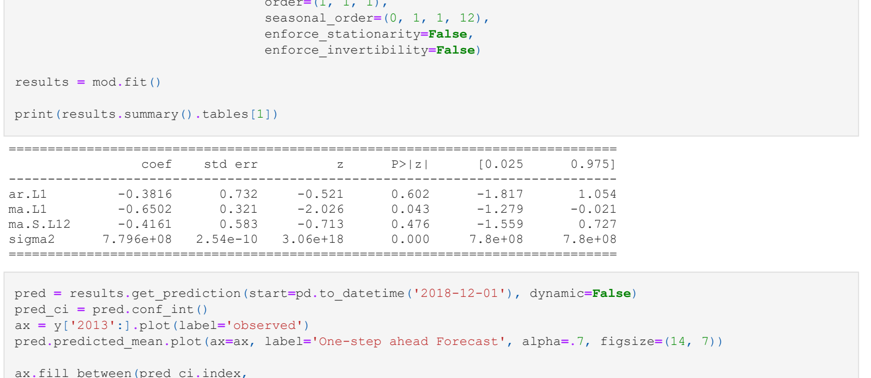
In [379…

```python
import statsmodels.api as sm
mod = sm.tsa.statespace.SARIMAX(y,
                                order=(1, 1, 1),
                                seasonal_order=(0, 1, 1, 12),
                                enforce_stationarity=False,
                                enforce_invertibility=False)

results = mod.fit()

print(results.summary().tables[1])
```

```
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.3816      0.732     -0.521      0.602      -1.817       1.054
ma.L1         -0.6502      0.321     -2.026      0.043      -1.279      -0.021
ma.S.L12      -0.4161      0.583     -0.713      0.476      -1.559       0.727
sigma2      7.796e+08   2.54e-10   3.06e+18      0.000     7.8e+08     7.8e+08
==============================================================================
```

In [380…

```python
pred = results.get_prediction(start=pd.to_datetime('2018-12-01'), dynamic=False)
pred_ci = pred.conf_int()
ax = y['2013':].plot(label='observed')
pred.predicted_mean.plot(ax=ax, label='One-step ahead Forecast', alpha=.7, figsize=(14, 7))

ax.fill_between(pred_ci.index,
 pred_ci.iloc[:, 0],
 pred_ci.iloc[:, 1], color='k', alpha=.2)
ax.set_xlabel('Date',fontsize=15)
ax.set_ylabel('Sales in Quantities',fontsize=15)
ax.set_title('The amount of Sales in Antwerp area',fontsize=20)
plt.legend()
plt.show()
```



In [381…

```python
y_forecasted = pred.predicted_mean
y_truth = y['2018-12-01':]
# Compute the mean squared error
mse = ((y_forecasted - y_truth) ** 2).mean()
print('The Mean Squared Error of our forecasts is {}'.format(round(mse, 2)))
print('The Root Mean Squared Error of our forecasts is {}'.format(round(np.sqrt(mse), 2)))
```

```
The Mean Squared Error of our forecasts is 1977781571.65
The Root Mean Squared Error of our forecasts is 44472.26
```

In [382…

```python
pred_uc = results.get_forecast(steps=100)
pred_ci = pred_uc.conf_int()
ax = y.plot(label='Observed', figsize=(20, 10),color='r',linewidth=6)
pred_uc.predicted_mean.plot(ax=ax, label='Forecast',color='dodgerblue',linewidth=6)
ax.set_xlabel('Date',fontsize=15)
ax.set_ylabel('Sales in Quantities',fontsize=15)
ax.set_title('The amount of Sales in Antwerp area',fontsize=20)
plt.legend()
plt.show()
```

Luxembourg Area

In [383…

```python
df=df1
```

In [384…

```python
df =df.loc[(df["Pharmacy Postcode (2)"] >=66) & (df["Pharmacy Postcode (2)"] <=69)]
```

In [385…

```python
df = df.sort_values('Delivery date')
```

In [386…

```python
df = df.groupby('Delivery date')['Units'].count().reset_index()
```

In [387…

```python
df = df.set_index('Delivery date')
```

In [388…

```python
y= df['Units'].resample('MS').mean()
```

In [389…

```python
import itertools
p = d = q = range(0, 2)

# Generate all different combinations of p, q and q triplets
pdq = list(itertools.product(p, d, q))

# Generate all different combinations of seasonal p, q and q triplets
seasonal_pdq = [(x[0], x[1], x[2], 12) for x in list(itertools.product(p, d, q))]

print('Examples of parameter combinations for Seasonal ARIMA...')
print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[1]))
print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[2]))
print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[3]))
print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[4]))
```

```
Examples of parameter combinations for Seasonal ARIMA...
SARIMAX: (0, 0, 1) x (0, 0, 1, 12)
SARIMAX: (0, 0, 1) x (0, 1, 0, 12)
SARIMAX: (0, 1, 0) x (0, 1, 1, 12)
SARIMAX: (0, 1, 0) x (1, 0, 0, 12)
```

In [390…

```python
for param in pdq:
    for param_seasonal in seasonal_pdq:
        try:
            mod = sm.tsa.statespace.SARIMAX(y,
                                            order=param,
                                            seasonal_order=param_seasonal,
                                            enforce_stationarity=False,
                                            enforce_invertibility=False)

            results = mod.fit()

            print('ARIMA{}x{}12 - AIC:{}'.format(param, param_seasonal, results.aic))
        except:
            continue
```

```
ARIMA(0, 0, 0)x(0, 0, 0, 12)12 - AIC:492.85359235522037
ARIMA(0, 0, 0)x(0, 0, 1, 12)12 - AIC:312.88461675307315
ARIMA(0, 0, 0)x(0, 1, 0, 12)12 - AIC:216.39880006994744
ARIMA(0, 0, 0)x(0, 1, 1, 12)12 - AIC:94.45584037543654
ARIMA(0, 0, 0)x(1, 0, 0, 12)12 - AIC:225.30003597410087
ARIMA(0, 0, 0)x(1, 0, 1, 12)12 - AIC:211.9394914759077
ARIMA(0, 0, 0)x(1, 1, 0, 12)12 - AIC:101.94805955694162
ARIMA(0, 0, 0)x(1, 1, 1, 12)12 - AIC:96.45590779616711
ARIMA(0, 0, 1)x(0, 0, 0, 12)12 - AIC:439.6066960294487
ARIMA(0, 0, 1)x(0, 0, 1, 12)12 - AIC:277.22235985466875
ARIMA(0, 0, 1)x(0, 1, 0, 12)12 - AIC:200.89057576028603
ARIMA(0, 0, 1)x(0, 1, 1, 12)12 - AIC:81.84434389502306
ARIMA(0, 0, 1)x(1, 0, 0, 12)12 - AIC:223.03231167225763
ARIMA(0, 0, 1)x(1, 0, 1, 12)12 - AIC:198.20431607281134
ARIMA(0, 0, 1)x(1, 1, 0, 12)12 - AIC:103.92983091406686
ARIMA(0, 0, 1)x(1, 1, 1, 12)12 - AIC:87.11188967626205
ARIMA(0, 1, 0)x(0, 0, 0, 12)12 - AIC:334.0998746268496
ARIMA(0, 1, 0)x(0, 0, 1, 12)12 - AIC:211.15757421021286
ARIMA(0, 1, 0)x(0, 1, 0, 12)12 - AIC:206.80447208884746
ARIMA(0, 1, 0)x(0, 1, 1, 12)12 - AIC:93.69479530287465
ARIMA(0, 1, 0)x(1, 0, 0, 12)12 - AIC:210.6211613650527
ARIMA(0, 1, 0)x(1, 0, 1, 12)12 - AIC:202.96746170751769
ARIMA(0, 1, 0)x(1, 1, 0, 12)12 - AIC:102.27006167993706
ARIMA(0, 1, 0)x(1, 1, 1, 12)12 - AIC:95.68953966645822
ARIMA(0, 1, 1)x(0, 0, 0, 12)12 - AIC:325.58708804483297
ARIMA(0, 1, 1)x(0, 0, 1, 12)12 - AIC:199.3966978599076
ARIMA(0, 1, 1)x(0, 1, 0, 12)12 - AIC:194.60646448284913
ARIMA(0, 1, 1)x(0, 1, 1, 12)12 - AIC:84.76284795496156
ARIMA(0, 1, 1)x(1, 0, 0, 12)12 - AIC:206.9775508704848
ARIMA(0, 1, 1)x(1, 0, 1, 12)12 - AIC:191.3732125004161
ARIMA(0, 1, 1)x(1, 1, 0, 12)12 - AIC:99.1264183114885
ARIMA(0, 1, 1)x(1, 1, 1, 12)12 - AIC:84.98192919031676
ARIMA(1, 0, 0)x(0, 0, 0, 12)12 - AIC:345.83794554580476
ARIMA(1, 0, 0)x(0, 0, 1, 12)12 - AIC:221.94581909817003
ARIMA(1, 0, 0)x(0, 1, 0, 12)12 - AIC:210.49212815917238
ARIMA(1, 0, 0)x(0, 1, 1, 12)12 - AIC:96.44814108287478
ARIMA(1, 0, 0)x(1, 0, 0, 12)12 - AIC:212.55990898962082
ARIMA(1, 0, 0)x(1, 0, 1, 12)12 - AIC:213.1106913535442
ARIMA(1, 0, 0)x(1, 1, 0, 12)12 - AIC:96.4481145591235
ARIMA(1, 0, 0)x(1, 1, 1, 12)12 - AIC:98.4410112832084
ARIMA(1, 0, 1)x(0, 0, 0, 12)12 - AIC:337.16271514295374
ARIMA(1, 0, 1)x(0, 0, 1, 12)12 - AIC:208.66498446820944
ARIMA(1, 0, 1)x(0, 1, 0, 12)12 - AIC:200.23838053714476
ARIMA(1, 0, 1)x(0, 1, 1, 12)12 - AIC:83.70556928654501
ARIMA(1, 0, 1)x(1, 0, 0, 12)12 - AIC:205.14067600611352
ARIMA(1, 0, 1)x(1, 0, 1, 12)12 - AIC:199.4838644253359
ARIMA(1, 0, 1)x(1, 1, 0, 12)12 - AIC:97.7458589693725
ARIMA(1, 0, 1)x(1, 1, 1, 12)12 - AIC:87.76772471876306
ARIMA(1, 1, 0)x(0, 0, 0, 12)12 - AIC:335.64900514082933
ARIMA(1, 1, 0)x(0, 0, 1, 12)12 - AIC:211.56131961536684
ARIMA(1, 1, 0)x(0, 1, 0, 12)12 - AIC:204.64048115103134
ARIMA(1, 1, 0)x(0, 1, 1, 12)12 - AIC:93.42885722234678
ARIMA(1, 1, 0)x(1, 0, 0, 12)12 - AIC:200.23984834005734
ARIMA(1, 1, 0)x(1, 0, 1, 12)12 - AIC:201.54880899281085
ARIMA(1, 1, 0)x(1, 1, 0, 12)12 - AIC:93.42915182992924
ARIMA(1, 1, 0)x(1, 1, 1, 12)12 - AIC:95.40174921300891
ARIMA(1, 1, 1)x(0, 0, 0, 12)12 - AIC:320.1131786490133
ARIMA(1, 1, 1)x(0, 0, 1, 12)12 - AIC:199.7623405845217
ARIMA(1, 1, 1)x(0, 1, 0, 12)12 - AIC:196.0026513229144
ARIMA(1, 1, 1)x(0, 1, 1, 12)12 - AIC:86.56649832112491
ARIMA(1, 1, 1)x(1, 0, 0, 12)12 - AIC:200.7765253524011
ARIMA(1, 1, 1)x(1, 0, 1, 12)12 - AIC:192.4568847522556
ARIMA(1, 1, 1)x(1, 1, 0, 12)12 - AIC:92.72924757293033
ARIMA(1, 1, 1)x(1, 1, 1, 12)12 - AIC:86.94300342036125
```

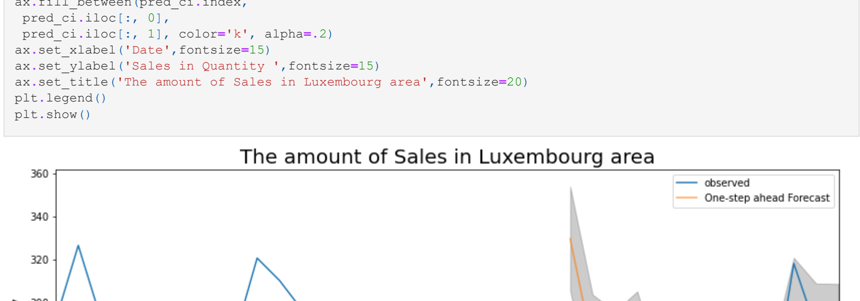
In [391…

```python
import statsmodels.api as sm
mod = sm.tsa.statespace.SARIMAX(y,
                                order=(0, 0, 1),
                                seasonal_order=(0, 1, 1, 12),
                                enforce_stationarity=False,
                                enforce_invertibility=False)

results = mod.fit()

print(results.summary().tables[1])
```

```
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ma.L1         -0.5084      0.193     -2.629      0.009      -0.888      -0.129
ma.S.L12      -1.0000   4757.255     -0.000      1.000   -9325.048    9323.048
sigma2        76.2745   3.63e+05      0.000      1.000   -7.11e+05    7.11e+05
==============================================================================
```

In [392…

```python
pred = results.get_prediction(start=pd.to_datetime('2018-12-01'), dynamic=False)
pred_ci = pred.conf_int()
ax = y['2013':].plot(label='observed')
pred.predicted_mean.plot(ax=ax, label='One-step ahead Forecast', alpha=.7, figsize=(14, 7))

ax.fill_between(pred_ci.index,
 pred_ci.iloc[:, 0],
 pred_ci.iloc[:, 1], color='k', alpha=.2)
ax.set_xlabel('Date',fontsize=15)
ax.set_ylabel('Sales in Quantity ',fontsize=15)
ax.set_title('The amount of Sales in Luxembourg area',fontsize=20)
plt.legend()
plt.show()
```



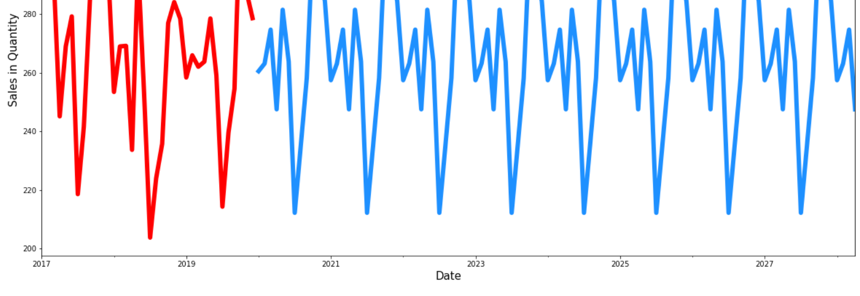
In [393…

```python
y_forecasted = pred.predicted_mean
y_truth = y['2018-12-01':]
# Compute the mean squared error
mse = ((y_forecasted - y_truth) ** 2).mean()
print('The Mean Squared Error of our forecasts is {}'.format(round(mse, 2)))
print('The Root Mean Squared Error of our forecasts is {}'.format(round(np.sqrt(mse), 2)))
```

```
The Mean Squared Error of our forecasts is 328.95
The Root Mean Squared Error of our forecasts is 18.14
```

In [394…

```python
pred_uc = results.get_forecast(steps=100)
pred_ci = pred_uc.conf_int()
ax = y.plot(label='Observed', figsize=(20, 10),color='r',linewidth=6)
pred_uc.predicted_mean.plot(ax=ax, label='Forecast',color='dodgerblue',linewidth=6)
ax.set_xlabel('Date',fontsize=15)
ax.set_ylabel('Sales in Quantity ',fontsize=15)
ax.set_title('The amount of Sales in Luxembourg area',fontsize=20)
plt.legend()
plt.show()
```



Flemish Region

In [251…

```python
df=df1
```

In [252…

```python
df =df.loc[(df["Pharmacy Postcode (2)"] >=30) & (df["Pharmacy Postcode (2)"] <=34)]
```

In [253…

```python
df = df.sort_values('Delivery date')
```

In [254…

```python
df = df.groupby('Delivery date')['Units'].count().reset_index()
```

In [255…

```python
df = df.set_index('Delivery date')
```

In [256…

```python
y= df['Units'].resample('MS').mean()
```

In [257…

```python
import itertools
p = d = q = range(0, 2)

# Generate all different combinations of p, q and q triplets
pdq = list(itertools.product(p, d, q))

# Generate all different combinations of seasonal p, q and q triplets
seasonal_pdq = [(x[0], x[1], x[2], 12) for x in list(itertools.product(p, d, q))]

print('Examples of parameter combinations for Seasonal ARIMA...')
print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[1]))
print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[2]))
print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[3]))
print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[4]))
```

```
Examples of parameter combinations for Seasonal ARIMA...
SARIMAX: (0, 0, 1) x (0, 0, 1, 12)
SARIMAX: (0, 0, 1) x (0, 1, 0, 12)
SARIMAX: (0, 1, 0) x (0, 1, 1, 12)
SARIMAX: (0, 1, 0) x (1, 0, 0, 12)
```

In [258…

```python
for param in pdq:
    for param_seasonal in seasonal_pdq:
        try:
            mod = sm.tsa.statespace.SARIMAX(y,
                                            order=param,
                                            seasonal_order=param_seasonal,
                                            enforce_stationarity=False,
                                            enforce_invertibility=False)

            results = mod.fit()

            print('ARIMA{}x{}12 - AIC:{}'.format(param, param_seasonal, results.aic))
        except:
            continue
```

```
ARIMA(0, 0, 0)x(0, 0, 0, 12)12 - AIC:618.0467478488881
ARIMA(0, 0, 0)x(0, 0, 1, 12)12 - AIC:383.61793411719333
ARIMA(0, 0, 0)x(0, 1, 0, 12)12 - AIC:308.20244665590764
ARIMA(0, 0, 0)x(0, 1, 1, 12)12 - AIC:152.76189398629234
ARIMA(0, 0, 0)x(1, 0, 0, 12)12 - AIC:317.25912955391726
ARIMA(0, 0, 0)x(1, 0, 1, 12)12 - AIC:300.62310740221625
ARIMA(0, 0, 0)x(1, 1, 0, 12)12 - AIC:164.86411706664006
ARIMA(0, 0, 0)x(1, 1, 1, 12)12 - AIC:150.55556771344112
ARIMA(0, 0, 1)x(0, 0, 0, 12)12 - AIC:558.6698978946073
ARIMA(0, 0, 1)x(0, 0, 1, 12)12 - AIC:345.45691703085305
ARIMA(0, 0, 1)x(0, 1, 0, 12)12 - AIC:294.33410366647564
ARIMA(0, 0, 1)x(0, 1, 1, 12)12 - AIC:139.71605729907606
ARIMA(0, 0, 1)x(1, 0, 0, 12)12 - AIC:316.9938793879542
ARIMA(0, 0, 1)x(1, 0, 1, 12)12 - AIC:288.91959156507266
ARIMA(0, 0, 1)x(1, 1, 0, 12)12 - AIC:158.92746024013178
ARIMA(0, 0, 1)x(1, 1, 1, 12)12 - AIC:133.3717713283507
ARIMA(0, 1, 0)x(0, 0, 0, 12)12 - AIC:431.00193167463
ARIMA(0, 1, 0)x(0, 0, 1, 12)12 - AIC:284.1404468497115
ARIMA(0, 1, 0)x(0, 1, 0, 12)12 - AIC:290.4947800796603
ARIMA(0, 1, 0)x(0, 1, 1, 12)12 - AIC:133.65589360963673
ARIMA(0, 1, 0)x(1, 0, 0, 12)12 - AIC:295.39092961384006
ARIMA(0, 1, 0)x(1, 0, 1, 12)12 - AIC:283.1734135242368
ARIMA(0, 1, 0)x(1, 1, 0, 12)12 - AIC:143.12646166796387
ARIMA(0, 1, 0)x(1, 1, 1, 12)12 - AIC:131.23191312171912
ARIMA(0, 1, 1)x(0, 0, 0, 12)12 - AIC:420.12176885408263
ARIMA(0, 1, 1)x(0, 0, 1, 12)12 - AIC:273.0952364445121
ARIMA(0, 1, 1)x(0, 1, 0, 12)12 - AIC:272.36563896604946
ARIMA(0, 1, 1)x(0, 1, 1, 12)12 - AIC:118.60571971407249
ARIMA(0, 1, 1)x(1, 0, 0, 12)12 - AIC:293.4177029979912
ARIMA(0, 1, 1)x(1, 0, 1, 12)12 - AIC:267.6416238938777
ARIMA(0, 1, 1)x(1, 1, 0, 12)12 - AIC:144.00375139171186
ARIMA(0, 1, 1)x(1, 1, 1, 12)12 - AIC:119.24181915801232
ARIMA(1, 0, 0)x(0, 0, 0, 12)12 - AIC:445.13178946751236
ARIMA(1, 0, 0)x(0, 0, 1, 12)12 - AIC:297.29452317988495
ARIMA(1, 0, 0)x(0, 1, 0, 12)12 - AIC:302.7201678024095
ARIMA(1, 0, 0)x(0, 1, 1, 12)12 - AIC:148.0228346582823
ARIMA(1, 0, 0)x(1, 0, 0, 12)12 - AIC:296.73517976343027
ARIMA(1, 0, 0)x(1, 0, 1, 12)12 - AIC:295.9168990575221
ARIMA(1, 0, 0)x(1, 1, 0, 12)12 - AIC:145.0506806238241
ARIMA(1, 0, 0)x(1, 1, 1, 12)12 - AIC:145.70303437367173
ARIMA(1, 0, 1)x(0, 0, 0, 12)12 - AIC:434.1854101069826
ARIMA(1, 0, 1)x(0, 0, 1, 12)12 - AIC:286.41292385406007
ARIMA(1, 0, 1)x(0, 1, 0, 12)12 - AIC:285.1609878532184
ARIMA(1, 0, 1)x(0, 1, 1, 12)12 - AIC:131.8743327257215
ARIMA(1, 0, 1)x(1, 0, 0, 12)12 - AIC:293.5499430529538
ARIMA(1, 0, 1)x(1, 0, 1, 12)12 - AIC:279.72489275583126
ARIMA(1, 0, 1)x(1, 1, 0, 12)12 - AIC:146.7149937310033
ARIMA(1, 0, 1)x(1, 1, 1, 12)12 - AIC:133.02972796133224
ARIMA(1, 1, 0)x(0, 0, 0, 12)12 - AIC:432.89691015696
ARIMA(1, 1, 0)x(0, 0, 1, 12)12 - AIC:284.8494728591838
ARIMA(1, 1, 0)x(0, 1, 0, 12)12 - AIC:282.7279507835501
ARIMA(1, 1, 0)x(0, 1, 1, 12)12 - AIC:131.93213616461588
ARIMA(1, 1, 0)x(1, 0, 0, 12)12 - AIC:280.42975706952785
ARIMA(1, 1, 0)x(1, 0, 1, 12)12 - AIC:280.86644806891866
ARIMA(1, 1, 0)x(1, 1, 0, 12)12 - AIC:131.0108288450655
ARIMA(1, 1, 0)x(1, 1, 1, 12)12 - AIC:132.98075812886304
ARIMA(1, 1, 1)x(0, 0, 0, 12)12 - AIC:419.7608089571791
ARIMA(1, 1, 1)x(0, 0, 1, 12)12 - AIC:274.4739215674124
ARIMA(1, 1, 1)x(0, 1, 0, 12)12 - AIC:272.0274139749728
ARIMA(1, 1, 1)x(0, 1, 1, 12)12 - AIC:118.58650137364342
ARIMA(1, 1, 1)x(1, 0, 0, 12)12 - AIC:282.75746656394796
ARIMA(1, 1, 1)x(1, 0, 1, 12)12 - AIC:269.616822820223
ARIMA(1, 1, 1)x(1, 1, 0, 12)12 - AIC:132.32511341186915
ARIMA(1, 1, 1)x(1, 1, 1, 12)12 - AIC:119.9625231246365
```

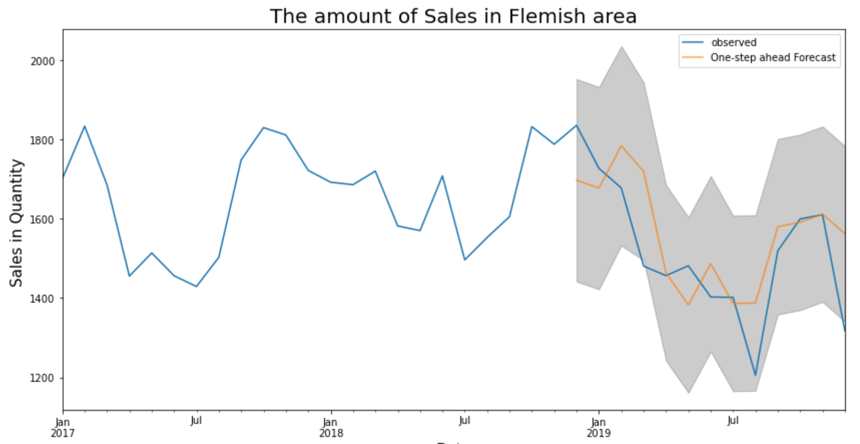
In [259…

```python
import statsmodels.api as sm
mod = sm.tsa.statespace.SARIMAX(y,
                                order=(1, 1, 1),
                                seasonal_order=(0, 1, 1, 12),
                                enforce_stationarity=False,
                                enforce_invertibility=False)

results = mod.fit()

print(results.summary().tables[1])
```

```
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.6240      0.658     -0.948      0.343      -1.914       0.666
ma.L1         -0.3422      0.860     -0.398      0.691      -2.027       1.343
ma.S.L12      -1.0011    562.727     -0.002      0.999   -1103.926    1101.924
sigma2      8512.8769   4.79e+06      0.002      0.999   -9.38e+06     9.4e+06
==============================================================================
```

In [260…

```python
pred = results.get_prediction(start=pd.to_datetime('2018-12-01'), dynamic=False)
pred_ci = pred.conf_int()
ax = y['2013':].plot(label='observed')
pred.predicted_mean.plot(ax=ax, label='One-step ahead Forecast', alpha=.7, figsize=(14, 7))

ax.fill_between(pred_ci.index,
 pred_ci.iloc[:, 0],
 pred_ci.iloc[:, 1], color='k', alpha=.2)
ax.set_xlabel('Date',fontsize=15)
ax.set_ylabel('Sales in Quantity ',fontsize=15)
ax.set_title('The amount of Sales in Flemish area',fontsize=20)
plt.legend()
plt.show()
```



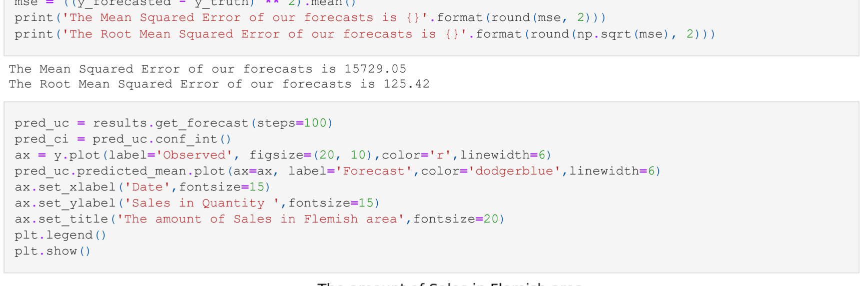
In [261…

```python
y_forecasted = pred.predicted_mean
y_truth = y['2018-12-01':]
# Compute the mean squared error
mse = ((y_forecasted - y_truth) ** 2).mean()
print('The Mean Squared Error of our forecasts is {}'.format(round(mse, 2)))
print('The Root Mean Squared Error of our forecasts is {}'.format(round(np.sqrt(mse), 2)))
```

```
The Mean Squared Error of our forecasts is 15729.05
The Root Mean Squared Error of our forecasts is 125.42
```

In [262…

```python
pred_uc = results.get_forecast(steps=100)
pred_ci = pred_uc.conf_int()
ax = y.plot(label='Observed', figsize=(20, 10),color='r',linewidth=6)
pred_uc.predicted_mean.plot(ax=ax, label='Forecast',color='dodgerblue',linewidth=6)
ax.set_xlabel('Date',fontsize=15)
ax.set_ylabel('Sales in Quantity ',fontsize=15)
ax.set_title('The amount of Sales in Flemish area',fontsize=20)
plt.legend()
plt.show()
```



Wallon Region

In [237…

```python
df=df1
```

In [238…

```python
df =df.loc[(df["Pharmacy Postcode (2)"] >=13) & (df["Pharmacy Postcode (2)"] <=14)]
```

In [239…

```python
df = df.sort_values('Delivery date')
```

In [240…

```python
df = df.groupby('Delivery date')['Units'].count().reset_index()
```

In [241…

```python
df = df.set_index('Delivery date')
```

In [243…

```python
y= df['Units'].resample('MS').mean()
```

In [244…

```python
import itertools
p = d = q = range(0, 2)

# Generate all different combinations of p, q and q triplets
pdq = list(itertools.product(p, d, q))

# Generate all different combinations of seasonal p, q and q triplets
seasonal_pdq = [(x[0], x[1], x[2], 12) for x in list(itertools.product(p, d, q))]

print('Examples of parameter combinations for Seasonal ARIMA...')
print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[1]))
print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[2]))
print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[3]))
print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[4]))
```

```
Examples of parameter combinations for Seasonal ARIMA...
SARIMAX: (0, 0, 1) x (0, 0, 1, 12)
SARIMAX: (0, 0, 1) x (0, 1, 0, 12)
SARIMAX: (0, 1, 0) x (0, 1, 1, 12)
SARIMAX: (0, 1, 0) x (1, 0, 0, 12)
```

In [245…

```python
for param in pdq:
    for param_seasonal in seasonal_pdq:
        try:
            mod = sm.tsa.statespace.SARIMAX(y,
                                            order=param,
                                            seasonal_order=param_seasonal,
                                            enforce_stationarity=False,
                                            enforce_invertibility=False)

            results = mod.fit()

            print('ARIMA{}x{}12 - AIC:{}'.format(param, param_seasonal, results.aic))
        except:
            continue
```

```
ARIMA(0, 0, 0)x(0, 0, 0, 12)12 - AIC:516.1988627719727
ARIMA(0, 0, 0)x(0, 0, 1, 12)12 - AIC:331.1491553757075
ARIMA(0, 0, 0)x(0, 1, 0, 12)12 - AIC:252.15294604286257
ARIMA(0, 0, 0)x(0, 1, 1, 12)12 - AIC:109.7316418682494
ARIMA(0, 0, 0)x(1, 0, 0, 12)12 - AIC:253.20224167578613
ARIMA(0, 0, 0)x(1, 0, 1, 12)12 - AIC:242.5535833632053
ARIMA(0, 0, 0)x(1, 1, 0, 12)12 - AIC:120.35135240636349
ARIMA(0, 0, 0)x(1, 1, 1, 12)12 - AIC:111.23757523426416
ARIMA(0, 0, 1)x(0, 0, 0, 12)12 - AIC:463.3096126184648
ARIMA(0, 0, 1)x(0, 0, 1, 12)12 - AIC:293.2075842532061
ARIMA(0, 0, 1)x(0, 1, 0, 12)12 - AIC:231.2857368087276
ARIMA(0, 0, 1)x(0, 1, 1, 12)12 - AIC:94.4517746557869
ARIMA(0, 0, 1)x(1, 0, 0, 12)12 - AIC:255.76231485575386
ARIMA(0, 0, 1)x(1, 0, 1, 12)12 - AIC:231.39896238621657
ARIMA(0, 0, 1)x(1, 1, 0, 12)12 - AIC:116.20740174479042
ARIMA(0, 0, 1)x(1, 1, 1, 12)12 - AIC:96.75200412045996
ARIMA(0, 1, 0)x(0, 0, 0, 12)12 - AIC:355.53528705676075
ARIMA(0, 1, 0)x(0, 0, 1, 12)12 - AIC:224.3314559665851
ARIMA(0, 1, 0)x(0, 1, 0, 12)12 - AIC:231.7499728498748
ARIMA(0, 1, 0)x(0, 1, 1, 12)12 - AIC:93.19573079102817
ARIMA(0, 1, 0)x(1, 0, 0, 12)12 - AIC:237.5815293449872
ARIMA(0, 1, 0)x(1, 0, 1, 12)12 - AIC:224.66329543336326
ARIMA(0, 1, 0)x(1, 1, 0, 12)12 - AIC:101.25304686789065
ARIMA(0, 1, 0)x(1, 1, 1, 12)12 - AIC:95.1883500696212
ARIMA(0, 1, 1)x(0, 0, 0, 12)12 - AIC:345.72302595260993
ARIMA(0, 1, 1)x(0, 0, 1, 12)12 - AIC:212.2836205908636
ARIMA(0, 1, 1)x(0, 1, 0, 12)12 - AIC:223.35905830389603
ARIMA(0, 1, 1)x(0, 1, 1, 12)12 - AIC:84.2973286098642
ARIMA(0, 1, 1)x(1, 0, 0, 12)12 - AIC:234.14041415495882
ARIMA(0, 1, 1)x(1, 0, 1, 12)12 - AIC:213.33359487220878
ARIMA(0, 1, 1)x(1, 1, 0, 12)12 - AIC:98.94946264502512
ARIMA(0, 1, 1)x(1, 1, 1, 12)12 - AIC:85.0275215482829
ARIMA(1, 0, 0)x(0, 0, 0, 12)12 - AIC:367.898301232108
ARIMA(1, 0, 0)x(0, 0, 1, 12)12 - AIC:235.48255876494173
ARIMA(1, 0, 0)x(0, 1, 0, 12)12 - AIC:239.9269933880727
ARIMA(1, 0, 0)x(0, 1, 1, 12)12 - AIC:101.46503258230007
ARIMA(1, 0, 0)x(1, 0, 0, 12)12 - AIC:239.49832571552702
ARIMA(1, 0, 0)x(1, 0, 1, 12)12 - AIC:235.7109330393729
ARIMA(1, 0, 0)x(1, 1, 0, 12)12 - AIC:101.46497747993659
ARIMA(1, 0, 0)x(1, 1, 1, 12)12 - AIC:103.1764109142354
ARIMA(1, 0, 1)x(0, 0, 0, 12)12 - AIC:354.79766640189433
ARIMA(1, 0, 1)x(0, 0, 1, 12)12 - AIC:220.36979558779973
ARIMA(1, 0, 1)x(0, 1, 0, 12)12 - AIC:231.873710283378
ARIMA(1, 0, 1)x(0, 1, 1, 12)12 - AIC:90.01860883104634
ARIMA(1, 0, 1)x(1, 0, 0, 12)12 - AIC:235.0566478802708
ARIMA(1, 0, 1)x(1, 0, 1, 12)12 - AIC:222.0002069940624
ARIMA(1, 0, 1)x(1, 1, 0, 12)12 - AIC:98.87388910001862
ARIMA(1, 0, 1)x(1, 1, 1, 12)12 - AIC:93.02079294496627
ARIMA(1, 1, 0)x(0, 0, 0, 12)12 - AIC:357.12297694801043
ARIMA(1, 1, 0)x(0, 0, 1, 12)12 - AIC:225.79824643476277
ARIMA(1, 1, 0)x(0, 1, 0, 12)12 - AIC:233.20354858800076
ARIMA(1, 1, 0)x(0, 1, 1, 12)12 - AIC:94.03613838284497
ARIMA(1, 1, 0)x(1, 0, 0, 12)12 - AIC:229.65356630941596
ARIMA(1, 1, 0)x(1, 0, 1, 12)12 - AIC:226.3354852220287
ARIMA(1, 1, 0)x(1, 1, 0, 12)12 - AIC:94.03610995085715
ARIMA(1, 1, 0)x(1, 1, 1, 12)12 - AIC:96.03601958268845
ARIMA(1, 1, 1)x(0, 0, 0, 12)12 - AIC:343.35446343455584
ARIMA(1, 1, 1)x(0, 0, 1, 12)12 - AIC:212.8334550721066
ARIMA(1, 1, 1)x(0, 1, 0, 12)12 - AIC:220.9211640458189
ARIMA(1, 1, 1)x(0, 1, 1, 12)12 - AIC:86.0650866487163
ARIMA(1, 1, 1)x(1, 0, 0, 12)12 - AIC:224.8352116868377
ARIMA(1, 1, 1)x(1, 0, 1, 12)12 - AIC:212.7587207604334
ARIMA(1, 1, 1)x(1, 1, 0, 12)12 - AIC:92.87142495444078
ARIMA(1, 1, 1)x(1, 1, 1, 12)12 - AIC:86.88520787615914
```

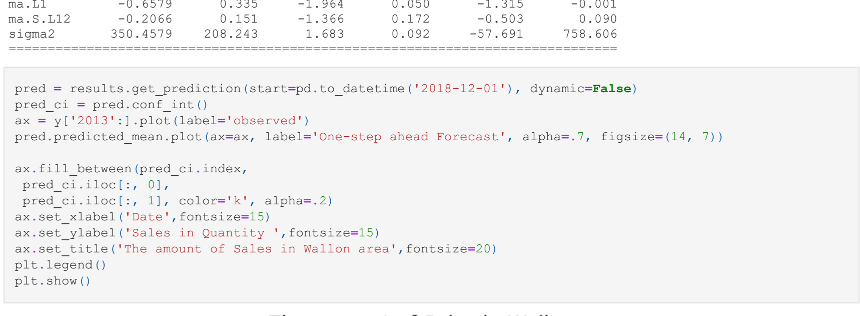
In [247…

```python
import statsmodels.api as sm
mod = sm.tsa.statespace.SARIMAX(y,
                                order=(0, 1, 1),
                                seasonal_order=(0, 1, 1, 12),
                                enforce_stationarity=False,
                                enforce_invertibility=False)

results = mod.fit()

print(results.summary().tables[1])
```

```
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ma.L1         -0.6579      0.335     -1.964      0.050      -1.315      -0.001
ma.S.L12      -0.2066      0.151     -1.366      0.172      -0.503       0.090
sigma2       350.4579    208.243      1.683      0.092     -57.691     758.606
==============================================================================
```

In [248…

```python
pred = results.get_prediction(start=pd.to_datetime('2018-12-01'), dynamic=False)
pred_ci = pred.conf_int()
ax = y['2013':].plot(label='observed')
pred.predicted_mean.plot(ax=ax, label='One-step ahead Forecast', alpha=.7, figsize=(14, 7))

ax.fill_between(pred_ci.index,
 pred_ci.iloc[:, 0],
 pred_ci.iloc[:, 1], color='k', alpha=.2)
ax.set_xlabel('Date',fontsize=15)
ax.set_ylabel('Sales in Quantity ',fontsize=15)
ax.set_title('The amount of Sales in Wallon area',fontsize=20)
plt.legend()
plt.show()
```



In [249…

```python
y_forecasted = pred.predicted_mean
y_truth = y['2018-12-01':]
# Compute the mean squared error
mse = ((y_forecasted - y_truth) ** 2).mean()
print('The Mean Squared Error of our forecasts is {}'.format(round(mse, 2)))
print('The Root Mean Squared Error of our forecasts is {}'.format(round(np.sqrt(mse), 2)))
```

```
The Mean Squared Error of our forecasts is 247.64
The Root Mean Squared Error of our forecasts is 15.74
```

In [250…

```python
pred_uc = results.get_forecast(steps=100)
pred_ci = pred_uc.conf_int()
ax = y.plot(label='Observed', figsize=(20, 10),color='r',linewidth=6)
pred_uc.predicted_mean.plot(ax=ax, label='Forecast',color='dodgerblue',linewidth=6)
ax.set_xlabel('Date',fontsize=15)
ax.set_ylabel('Sales in Quantity ',fontsize=15)
ax.set_title('The amount of Sales in Wallon area',fontsize=20)
plt.legend()
plt.show()
```

General Sales in Quantities

In [395…

```python
df=df1
```

In [396…

```python
df = df.sort_values('Delivery date')
```

In [397…

```python
df = df.groupby('Delivery date')['Units'].sum().reset_index()
```

In [398…

```python
df = df.set_index('Delivery date')
```

In [399…

```python
y= df['Units'].resample('MS').mean()
```

In [400…

```python
import itertools
p = d = q = range(0, 2)

# Generate all different combinations of p, q and q triplets
```

```python
pdq = list(itertools.product(p, d, q))

# Generate all different combinations of seasonal p, q and q triplets
seasonal_pdq = [(x[0], x[1], x[2], 12) for x in list(itertools.product(p, d, q))]

print('Examples of parameter combinations for Seasonal ARIMA...')
print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[1]))
print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[2]))
print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[3]))
print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[4]))
```

```
Examples of parameter combinations for Seasonal ARIMA...
SARIMAX: (0, 0, 1) x (0, 0, 1, 12)
SARIMAX: (0, 0, 1) x (0, 1, 0, 12)
SARIMAX: (0, 1, 0) x (0, 1, 1, 12)
SARIMAX: (0, 1, 0) x (1, 0, 0, 12)
```

In [401…

```python
for param in pdq:
    for param_seasonal in seasonal_pdq:
        try:
            mod = sm.tsa.statespace.SARIMAX(y,
                                            order=param,
                                            seasonal_order=param_seasonal,
                                            enforce_stationarity=False,
                                            enforce_invertibility=False)

            results = mod.fit()

            print('ARIMA{}x{}12 - AIC:{}'.format(param, param_seasonal, results.aic))
        except:
            continue
```

```
ARIMA(0, 0, 0)x(0, 0, 0, 12)12 - AIC:1063.5499302309097
ARIMA(0, 0, 0)x(0, 0, 1, 12)12 - AIC:1906.9803224267962
ARIMA(0, 0, 0)x(0, 1, 0, 12)12 - AIC:588.8890488566699
ARIMA(0, 0, 0)x(0, 1, 1, 12)12 - AIC:278.8662813344097
ARIMA(0, 0, 0)x(1, 0, 0, 12)12 - AIC:610.2703700723097
ARIMA(0, 0, 0)x(1, 0, 1, 12)12 - AIC:586.1678236194456
ARIMA(0, 0, 0)x(1, 1, 0, 12)12 - AIC:302.79357012695823
ARIMA(0, 0, 0)x(1, 1, 1, 12)12 - AIC:277.7323150887992
ARIMA(0, 0, 1)x(0, 0, 0, 12)12 - AIC:1010.6604788746681
ARIMA(0, 0, 1)x(0, 0, 1, 12)12 - AIC:653.7217066419591
ARIMA(0, 0, 1)x(0, 1, 0, 12)12 - AIC:559.7671207675
ARIMA(0, 0, 1)x(0, 1, 1, 12)12 - AIC:256.3564452910347
ARIMA(0, 0, 1)x(1, 0, 0, 12)12 - AIC:709.7357140825376
ARIMA(0, 0, 1)x(1, 0, 1, 12)12 - AIC:653.0718556410461
ARIMA(0, 0, 1)x(1, 1, 0, 12)12 - AIC:304.58918886697165
ARIMA(0, 0, 1)x(1, 1, 1, 12)12 - AIC:254.0612377465954
ARIMA(0, 1, 0)x(0, 0, 0, 12)12 - AIC:866.7570457594978
ARIMA(0, 1, 0)x(0, 0, 1, 12)12 - AIC:553.9312704734422
ARIMA(0, 1, 0)x(0, 1, 0, 12)12 - AIC:568.808917998308
ARIMA(0, 1, 0)x(0, 1, 1, 12)12 - AIC:259.71218128371925
ARIMA(0, 1, 0)x(1, 0, 0, 12)12 - AIC:577.1951271927386
ARIMA(0, 1, 0)x(1, 0, 1, 12)12 - AIC:555.5692181160414
ARIMA(0, 1, 0)x(1, 1, 0, 12)12 - AIC:284.9219818004791
ARIMA(0, 1, 0)x(1, 1, 1, 12)12 - AIC:259.20636458447774
ARIMA(0, 1, 1)x(0, 0, 0, 12)12 - AIC:842.4297588999764
ARIMA(0, 1, 1)x(0, 0, 1, 12)12 - AIC:526.315683915649
ARIMA(0, 1, 1)x(0, 1, 0, 12)12 - AIC:531.8195624776407
ARIMA(0, 1, 1)x(0, 1, 1, 12)12 - AIC:229.9092858606815
ARIMA(0, 1, 1)x(1, 0, 0, 12)12 - AIC:575.8206728199069
ARIMA(0, 1, 1)x(1, 0, 1, 12)12 - AIC:525.0722810325161
ARIMA(0, 1, 1)x(1, 1, 0, 12)12 - AIC:279.47738711416054
ARIMA(0, 1, 1)x(1, 1, 1, 12)12 - AIC:231.8718046923213
ARIMA(1, 0, 0)x(0, 0, 0, 12)12 - AIC:893.7617401725245
ARIMA(1, 0, 0)x(0, 0, 1, 12)12 - AIC:580.5894576271274
ARIMA(1, 0, 0)x(0, 1, 0, 12)12 - AIC:584.167908286048
ARIMA(1, 0, 0)x(0, 1, 1, 12)12 - AIC:280.83369134472105
ARIMA(1, 0, 0)x(1, 0, 0, 12)12 - AIC:581.6461648834567
ARIMA(1, 0, 0)x(1, 0, 1, 12)12 - AIC:582.2955906085525
ARIMA(1, 0, 0)x(1, 1, 0, 12)12 - AIC:279.93149788281175
ARIMA(1, 0, 0)x(1, 1, 1, 12)12 - AIC:279.16340903885765
ARIMA(1, 0, 1)x(0, 0, 0, 12)12 - AIC:869.0808144867456
ARIMA(1, 0, 1)x(0, 0, 1, 12)12 - AIC:549.2226631248564
ARIMA(1, 0, 1)x(0, 1, 0, 12)12 - AIC:561.0232504396487
ARIMA(1, 0, 1)x(0, 1, 1, 12)12 - AIC:254.9545797670638
ARIMA(1, 0, 1)x(1, 0, 0, 12)12 - AIC:577.9200211104396
ARIMA(1, 0, 1)x(1, 0, 1, 12)12 - AIC:550.901216384265
ARIMA(1, 0, 1)x(1, 1, 0, 12)12 - AIC:281.27138426864883
ARIMA(1, 0, 1)x(1, 1, 1, 12)12 - AIC:255.7813798725807
ARIMA(1, 1, 0)x(0, 0, 0, 12)12 - AIC:868.3733428565278
ARIMA(1, 1, 0)x(0, 0, 1, 12)12 - AIC:555.7962716773525
ARIMA(1, 1, 0)x(0, 1, 0, 12)12 - AIC:566.0314964848517
ARIMA(1, 1, 0)x(0, 1, 1, 12)12 - AIC:258.3937899934461
ARIMA(1, 1, 0)x(1, 0, 0, 12)12 - AIC:556.5157266580233
ARIMA(1, 1, 0)x(1, 0, 1, 12)12 - AIC:557.6483628488605
ARIMA(1, 1, 0)x(1, 1, 0, 12)12 - AIC:258.6850494304908
ARIMA(1, 1, 0)x(1, 1, 1, 12)12 - AIC:259.14844345925195
ARIMA(1, 1, 1)x(0, 0, 0, 12)12 - AIC:842.5700452315516
ARIMA(1, 1, 1)x(0, 0, 1, 12)12 - AIC:525.0870650752012
ARIMA(1, 1, 1)x(0, 1, 0, 12)12 - AIC:533.7965932303598
ARIMA(1, 1, 1)x(0, 1, 1, 12)12 - AIC:230.7842719647387
ARIMA(1, 1, 1)x(1, 0, 0, 12)12 - AIC:552.6397001761521
ARIMA(1, 1, 1)x(1, 0, 1, 12)12 - AIC:527.0041529530695
ARIMA(1, 1, 1)x(1, 1, 0, 12)12 - AIC:255.9910076023476
ARIMA(1, 1, 1)x(1, 1, 1, 12)12 - AIC:232.4216078080466
```

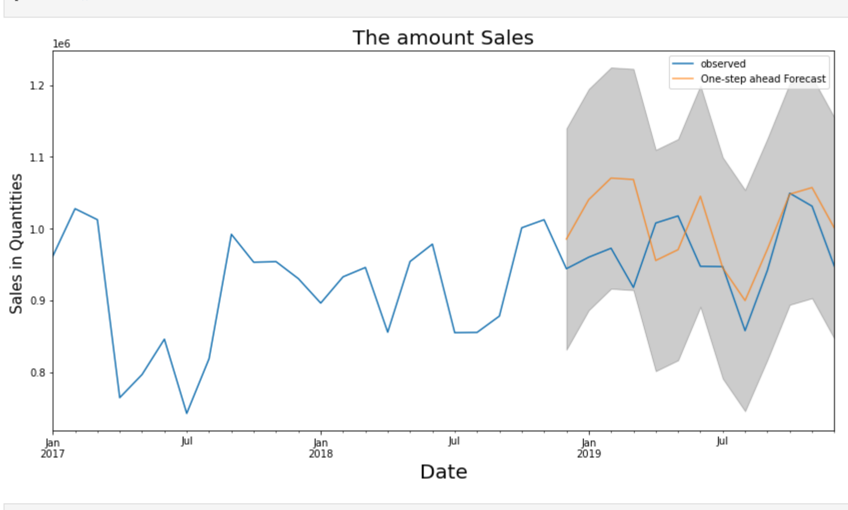
In [402…

```python
import statsmodels.api as sm
mod = sm.tsa.statespace.SARIMAX(y,
                                order=(1, 1, 1),
                                seasonal_order=(0, 1, 1, 12),
                                enforce_stationarity=False,
                                enforce_invertibility=False)

results = mod.fit()

print(results.summary().tables[1])
```

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.4525	0.872	-0.519	0.604	-2.161	1.256
ma.L1	-0.7543	0.537	-1.404	0.160	-1.807	0.299
ma.S.L12	-0.1154	0.944	-0.122	0.903	-1.965	1.734
sigma2	6.185e+09	6.45e-11	9.6e+19	0.000	6.19e+09	6.19e+09

In [403…

```python
pred = results.get_prediction(start=pd.to_datetime('2018-12-01'), dynamic=False)
pred_ci = pred.conf_int()
ax = y['2013':].plot(label='observed')
pred.predicted_mean.plot(ax=ax, label='One-step ahead Forecast', alpha=.7, figsize=(14, 7))

ax.fill_between(pred_ci.index,
 pred_ci.iloc[:, 0],
 pred_ci.iloc[:, 1], color='k', alpha=.2)
ax.set_xlabel('Date',fontsize=20)
ax.set_ylabel('Sales in Quantities ',fontsize=15)
ax.set_title('The amount Sales',fontsize=20)
plt.legend()
plt.show()
```



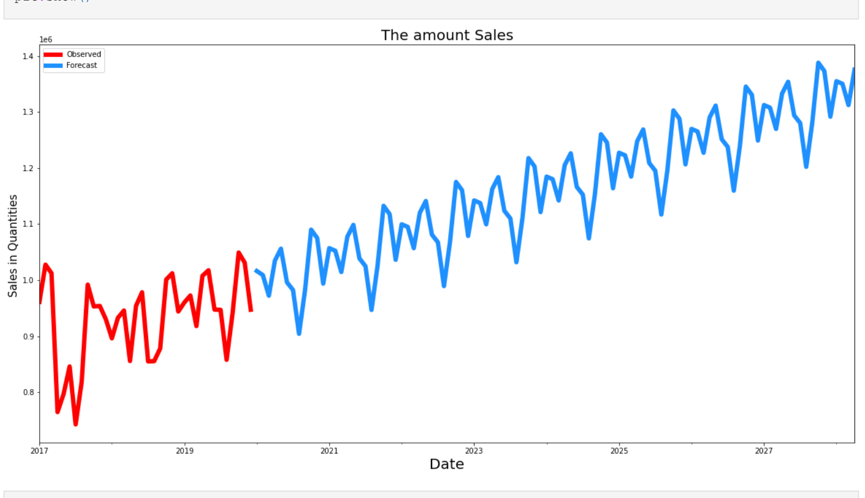
In [404…

```python
y_forecasted = pred.predicted_mean
y_truth = y['2018-12-01':]
# Compute the mean squared error
mse = ((y_forecasted - y_truth) ** 2).mean()
print('The Mean Squared Error of our forecasts is {}'.format(round(mse, 2)))
print('The Root Mean Squared Error of our forecasts is {}'.format(round(np.sqrt(mse), 2)))
```

```
The Mean Squared Error of our forecasts is 4689380469.27
The Root Mean Squared Error of our forecasts is 68479.05
```

In [405…

```python
pred_uc = results.get_forecast(steps=100)
pred_ci = pred_uc.conf_int()
ax = y.plot(label='Observed', figsize=(20, 10),color='r',linewidth=6)
pred_uc.predicted_mean.plot(ax=ax, label='Forecast',color='dodgerblue',linewidth=6)
ax.set_xlabel('Date',fontsize=20)
ax.set_ylabel('Sales in Quantities ',fontsize=15)
ax.set_title('The Amount of Sales',fontsize=20)
plt.legend()
plt.show()
```



In [ ]: