11-Forecasting-Exercises-Solutions

October 19, 2022

For more information, visit us at www.pieriandata.com

1 Forecasting Exercises - Solutions

This exercise walks through a SARIMA prediction and forecast similar to the one done on the Mauna Loa CO dataset. This time we're using a seasonal time series of California Hospitality Industry Employees.

IMPORTANT NOTE! Make sure you don't run the cells directly above the example output shown, otherwise you will end up writing over the example output!

[1]:
```
# RUN THIS CELL
import pandas as pd
import numpy as np
%matplotlib inline

# Load specific forecasting tools
from statsmodels.tsa.statespace.sarimax import SARIMAX

from statsmodels.graphics.tsaplots import plot_acf,plot_pacf # for determining␣
↪(p,q) orders
from statsmodels.tsa.seasonal import seasonal_decompose      # for ETS Plots
from pmdarima import auto_arima                              # for determining␣
↪ARIMA orders

# Load specific evaluation tools
from sklearn.metrics import mean_squared_error
from statsmodels.tools.eval_measures import rmse

# Ignore harmless warnings
import warnings
warnings.filterwarnings("ignore")
```

```
# Load datasets
df = pd.read_csv('../Data/HospitalityEmployees.
 ↪csv',index_col='Date',parse_dates=True)
df.index.freq = 'MS'
print(len(df))
print(df.head())
```

```
348
            Employees
Date
1990-01-01     1064.5
1990-02-01     1074.5
1990-03-01     1090.0
1990-04-01     1097.4
1990-05-01     1108.7
```

So df has 348 records and one column. The data represents the number of employees in thousands of persons as monthly averages from January, 1990 to December 2018.

1.0.1 1. Plot the source data

Create a line chart of the dataset. Optional: add a title and y-axis label.

[ ]:
```
## CODE HERE
```

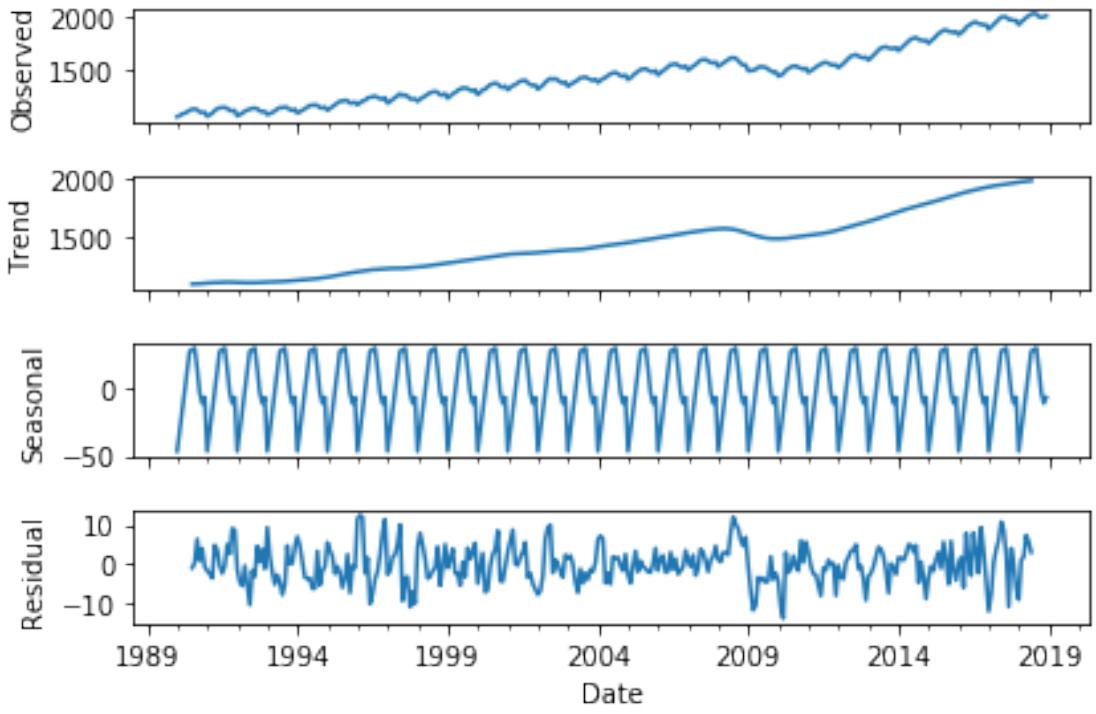
[2]:
```
# DON'T WRITE HERE
title='California Hospitality Industry Employees'
ylabel='Thousands of Persons'
xlabel='' # we don't really need a label here

ax = df['Employees'].plot(figsize=(12,5),title=title)
ax.autoscale(axis='x',tight=True)
ax.set(xlabel=xlabel, ylabel=ylabel);
```

1.0.2 2. Run an ETS Decomposition

Use an 'additive' model.

[ ]:

[3]:
```
# DON'T WRITE HERE
result = seasonal_decompose(df['Employees'], model='add')
result.plot();
```



1.0.3 3. Run pmdarima.auto_arima to obtain recommended orders

This may take awhile as there are a lot of combinations to evaluate.

[ ]:

[4]:
```
# DON'T WRITE HERE
auto_arima(df['Employees'],seasonal=True,m=12).summary()
```

[4]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                                Statespace Model Results
==========================================================================
==========
Dep. Variable:                                    y   No. Observations:
348
Model:             SARIMAX(0, 1, 0)x(2, 0, 0, 12)   Log Likelihood
-1134.664
Date:                            Wed, 27 Mar 2019   AIC
2277.328
Time:                                    13:12:10   BIC
2292.726
Sample:                                         0   HQIC
2283.459
                                              - 348
Covariance Type:                              opg
==========================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
--------------------------------------------------------------------------
intercept     -0.0477      0.292     -0.163      0.870      -0.620       0.524
ar.S.L12       0.5291      0.040     13.286      0.000       0.451       0.607
ar.S.L24       0.4303      0.041     10.453      0.000       0.350       0.511
sigma2        37.2952      2.157     17.294      0.000      33.068      41.522
==========================================================================
===
Ljung-Box (Q):                       99.53   Jarque-Bera (JB):
51.67
Prob(Q):                              0.00   Prob(JB):
0.00
Heteroskedasticity (H):               0.86   Skew:
-0.29
Prob(H) (two-sided):                  0.42   Kurtosis:
4.80
==========================================================================
===

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

You should see a recommended ARIMA Order of (0,1,0) combined with a seasonal order of (2,0,0,12). ### 4. Split the data into train/test sets Set one year (12 records) for testing. There is more than one way to do this!

[ ]:

[5]:
```
# DON'T WRITE HERE
train = df.iloc[:len(df)-12]
test = df.iloc[len(df)-12:]
```

1.0.4 5. Fit a SARIMA(0,1,0)(2,0,0,12) model to the training set

[ ]:
```
```

[6]:
```
# DON'T WRITE HERE
model = SARIMAX(train['Employees'],order=(0,1,0),seasonal_order=(2,0,0,12))
results = model.fit()
results.summary()
```

[6]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                                Statespace Model Results
==========================================================================================
Dep. Variable:                           Employees   No. Observations:
336
Model:             SARIMAX(0, 1, 0)x(2, 0, 0, 12)   Log Likelihood
-1095.407
Date:                            Wed, 27 Mar 2019   AIC
2196.814
Time:                                    13:12:24   BIC
2208.256
Sample:                                01-01-1990   HQIC
2201.375
                                     - 12-01-2017
Covariance Type:                               opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.S.L12       0.5204      0.040     13.051      0.000       0.442       0.599
ar.S.L24       0.4385      0.041     10.593      0.000       0.357       0.520
sigma2        37.1907      2.165     17.175      0.000      32.947      41.435
==============================================================================
===
Ljung-Box (Q):                      102.80   Jarque-Bera (JB):
56.66
Prob(Q):                              0.00   Prob(JB):
0.00
Heteroskedasticity (H):               1.06   Skew:
-0.35
Prob(H) (two-sided):                  0.74   Kurtosis:
4.89
```

```
==============================================================================
===

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

1.0.5 6. Obtain predicted values

[ ]:

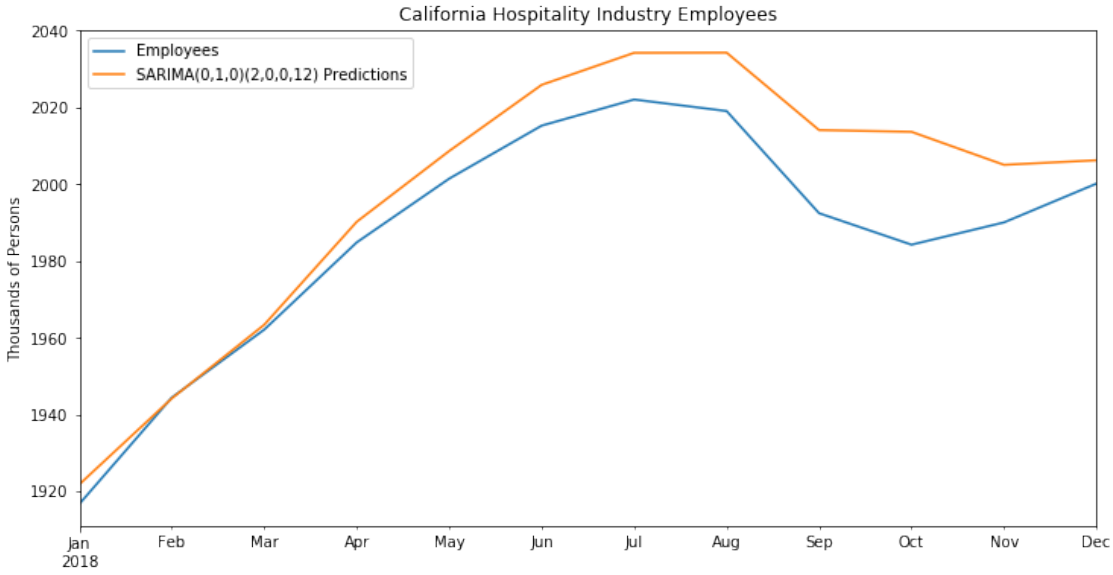
[7]:
```
# DON'T WRITE HERE
start=len(train)
end=len(train)+len(test)-1
predictions = results.predict(start=start, end=end, dynamic=False,␣
↪typ='levels').rename('SARIMA(0,1,0)(2,0,0,12) Predictions')
```

1.0.6 7. Plot predictions against known values

Optional: add a title and y-axis label.

[ ]:

[8]:
```
# DON'T WRITE HERE
title='California Hospitality Industry Employees'
ylabel='Thousands of Persons'
xlabel=''

ax = test['Employees'].plot(legend=True,figsize=(12,6),title=title)
predictions.plot(legend=True)
ax.autoscale(axis='x',tight=True)
ax.set(xlabel=xlabel, ylabel=ylabel);
```



1.0.7 8. Evaluate the Model using MSE and RMSE

You can run both from the same cell if you want.

[ ]:

[9]:
```
# DON'T WRITE HERE
error1 = mean_squared_error(test['Employees'], predictions)
error2 = rmse(test['Employees'], predictions)
print(f'SARIMA(0,1,0)(2,0,0,12) MSE Error: {error1:11.10}')
print(f'SARIMA(0,1,0)(2,0,0,12) RMSE Error: {error2:11.10}')
```

```
SARIMA(0,1,0)(2,0,0,12) MSE Error: 182.8506646
SARIMA(0,1,0)(2,0,0,12) RMSE Error: 13.52222854
```

1.0.8 9. Retrain the model on the full data and forecast one year into the future

[ ]:

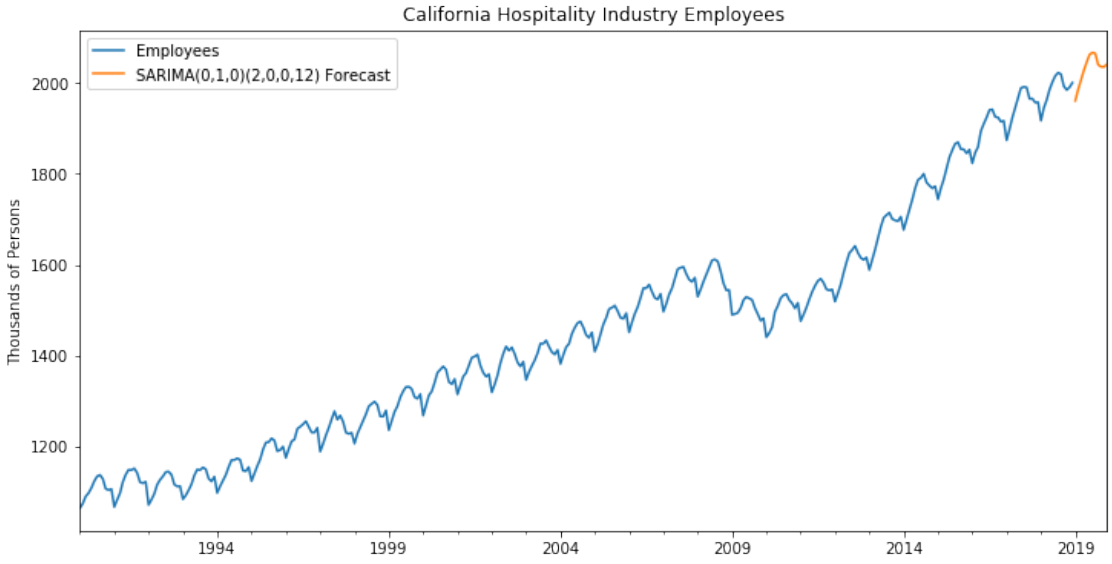
[10]:
```
# DON'T WRITE HERE
model = SARIMAX(df['Employees'],order=(0,1,0),seasonal_order=(2,0,0,12))
results = model.fit()
fcast = results.predict(len(df),len(df)+11,typ='levels').
 ↪rename('SARIMA(0,1,0)(2,0,0,12) Forecast')
```

1.0.9 10. Plot the forecasted values alongside the original data

Optional: add a title and y-axis label.

[ ]:

[11]:
```
# DON'T WRITE HERE
title='California Hospitality Industry Employees'
ylabel='Thousands of Persons'
xlabel=''

ax = df['Employees'].plot(legend=True,figsize=(12,6),title=title)
fcast.plot(legend=True)
ax.autoscale(axis='x',tight=True)
ax.set(xlabel=xlabel, ylabel=ylabel);
```



1.1 Great job!