Week14_Python_Code

November 15, 2021

1 Required Modules

pdarima

pip install pmdarima

[23]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from scipy import stats
import statsmodels.api as sm
from statsmodels.tsa.arima_model import ARIMA
import pmdarima as pm

from sklearn.metrics import mean_absolute_percentage_error

import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
```

2 Data Loading

[27]:
```
df = pd.read_sas("solarpv.sas7bdat")
```

3 Data Description

Time series frequency is *weekly*.

[28]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 42 entries, 0 to 41
Data columns (total 4 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   EDT          42 non-null     datetime64[ns]
 1   kW_Gen       42 non-null     float64
 2   Cloud_Cover  42 non-null     float64
```

```
 3   cosval       42 non-null     float64
dtypes: datetime64[ns](1), float64(3)
memory usage: 1.4 KB
```

[29]:
```
# First row
print(df.iloc[0])

# Last row
print(df.iloc[-1])
```

```
EDT            2014-10-05 00:00:00
kW_Gen                     0.55341
Cloud_Cover                   4.75
cosval                   -0.300642
Name: 0, dtype: object
EDT            2015-07-19 00:00:00
kW_Gen                    0.648016
Cloud_Cover               5.406013
cosval                    0.860451
Name: 41, dtype: object
```

Subset the `solarpv` dataset

[30]:
```
df = df[['EDT', 'kW_Gen']]

# Setting the idex as DateTime index object
df.set_index('EDT', inplace=True)

# Telling pandas to infer the frequency
df.index.freq = df.index.inferred_freq

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 42 entries, 2014-10-05 to 2015-07-19
Freq: W-SUN
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   kW_Gen  42 non-null     float64
dtypes: float64(1)
memory usage: 672.0 bytes
```

[31]:
```
df.head()
```

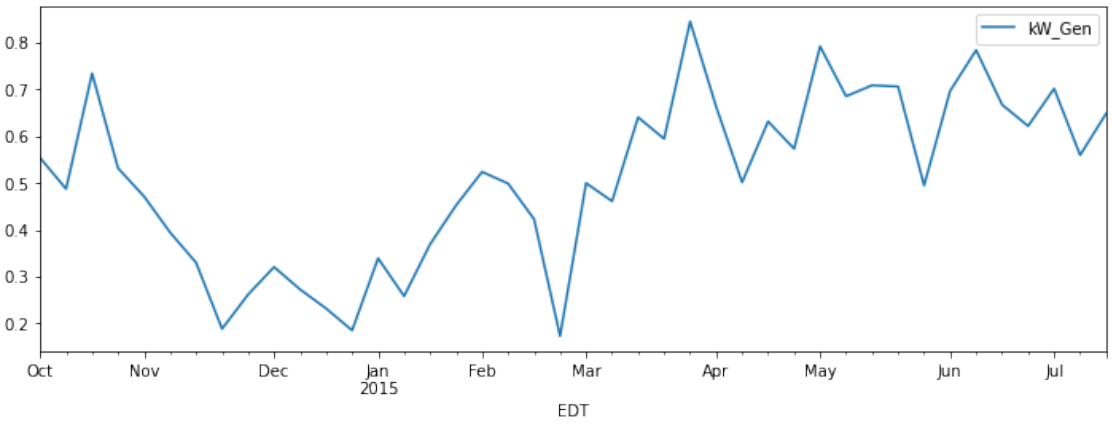
[31]:

EDT	kW_Gen
2014-10-05	0.553410
2014-10-12	0.487093

```
2014-10-19  0.733748
2014-10-26  0.531250
2014-11-02  0.471055
```

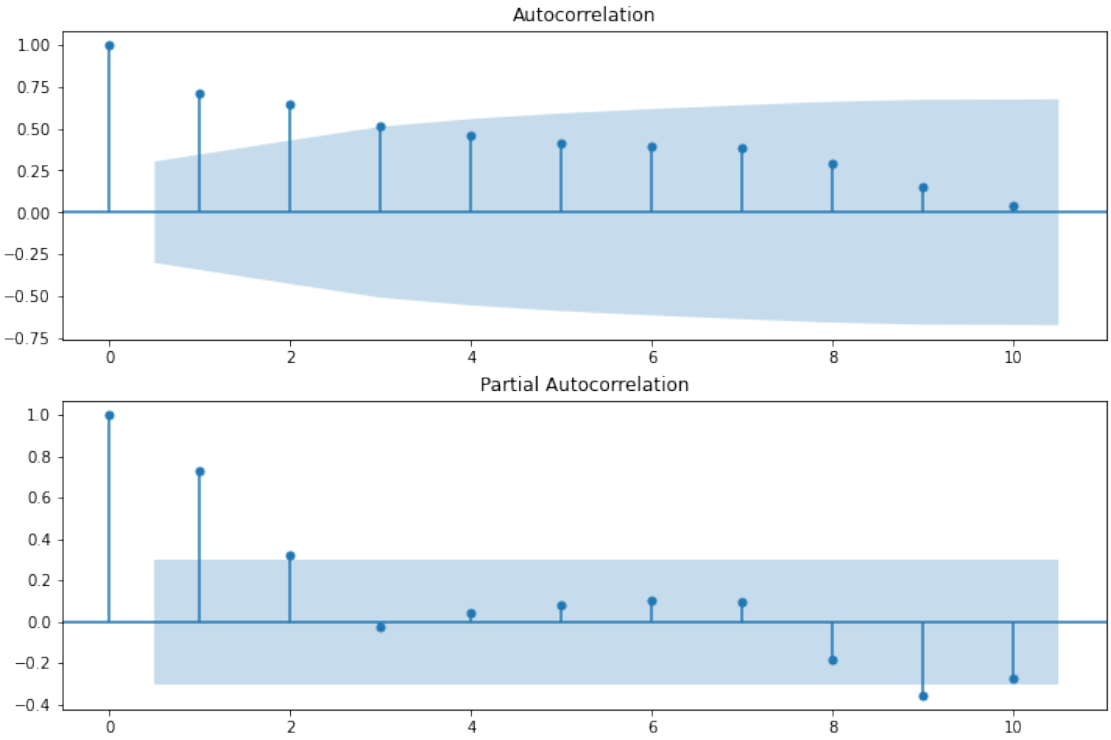
[32]:
```
df.plot(figsize=(12,4))
```

[32]:
```
<AxesSubplot:xlabel='EDT'>
```



4 ACF and PACF

[45]:
```
fig = plt.figure(figsize=(12,8))
ax1 = fig.add_subplot(211)
fig = sm.graphics.tsa.plot_acf(df, lags=10, ax=ax1)
ax2 = fig.add_subplot(212)
fig = sm.graphics.tsa.plot_pacf(df, lags=10, ax=ax2)
```



4.1 Ljung Box Test

[34]:
```
res = sm.tsa.ARMA(df["kW_Gen"], (1,1)).fit(disp=-1)
sm.stats.acorr_ljungbox(res.resid, lags=[5], return_df=True)
```

[34]:

	lb_stat	lb_pvalue
5	0.751124	0.980046

5 ARIMA

`auto_arima` uses a stepwise approach to search multiple combinations of `p`,`d`,`q` parameters and chooses the best model that has the least `AIC`.

[36]:
```
model = pm.auto_arima(df.kW_Gen, start_p=1, start_q=1,
                      test='adf',       # use adftest to find optimal 'd'
                      max_p=3, max_q=3, # maximum p and q
                      m=1,              # frequency of series
                      d=None,           # let model determine 'd'
                      seasonal=False,   # No Seasonality
                      start_P=0,
                      D=0,
                      trace=True,
                      error_action='ignore',
```

```
                      suppress_warnings=True,
                      stepwise=True)

print(model.summary())
```

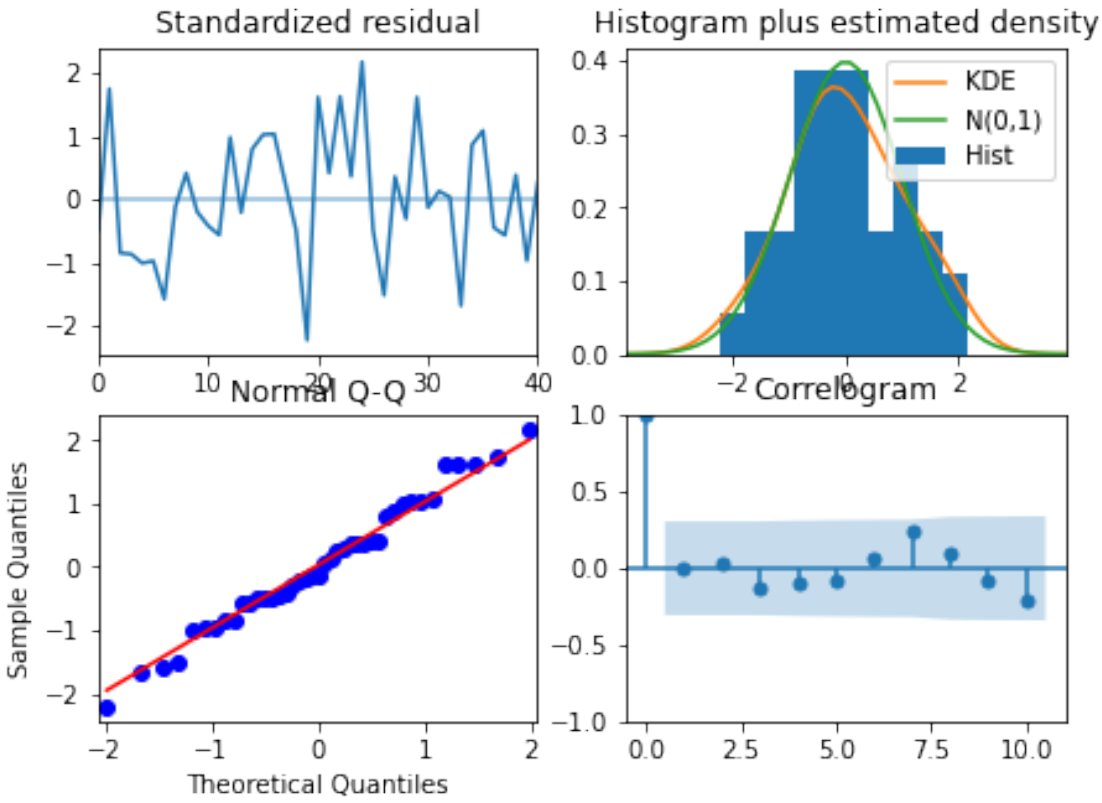
```
Performing stepwise search to minimize aic
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=-46.265, Time=0.08 sec
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=-42.879, Time=0.02 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=-47.479, Time=0.02 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=-48.191, Time=0.02 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=-44.867, Time=0.01 sec
 ARIMA(0,1,2)(0,0,0)[0] intercept   : AIC=-46.226, Time=0.04 sec
 ARIMA(1,1,2)(0,0,0)[0] intercept   : AIC=-44.275, Time=0.09 sec
 ARIMA(0,1,1)(0,0,0)[0]             : AIC=-50.154, Time=0.01 sec
 ARIMA(1,1,1)(0,0,0)[0]             : AIC=-48.213, Time=0.03 sec
 ARIMA(0,1,2)(0,0,0)[0]             : AIC=-48.183, Time=0.04 sec
 ARIMA(1,1,0)(0,0,0)[0]             : AIC=-49.456, Time=0.02 sec
 ARIMA(1,1,2)(0,0,0)[0]             : AIC=-46.241, Time=0.06 sec

Best model:  ARIMA(0,1,1)(0,0,0)[0]
Total fit time: 0.450 seconds
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                   42
Model:               SARIMAX(0, 1, 1)   Log Likelihood                27.077
Date:                Mon, 15 Nov 2021   AIC                            -50.154
Time:                        16:03:36   BIC                            -46.727
Sample:                             0   HQIC                           -48.906
                                  - 42
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ma.L1         -0.4511      0.153     -2.958      0.003      -0.750      -0.152
sigma2         0.0155      0.004      3.993      0.000       0.008       0.023
===================================================================================
===
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):
0.34
Prob(Q):                              0.99   Prob(JB):
0.85
Heteroskedasticity (H):               0.85   Skew:
0.06
Prob(H) (two-sided):                  0.77   Kurtosis:
2.58
===================================================================================
===
```

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

6 Diagnostics

[37]:
```
model.plot_diagnostics(figsize=(7,5))
plt.show()
```

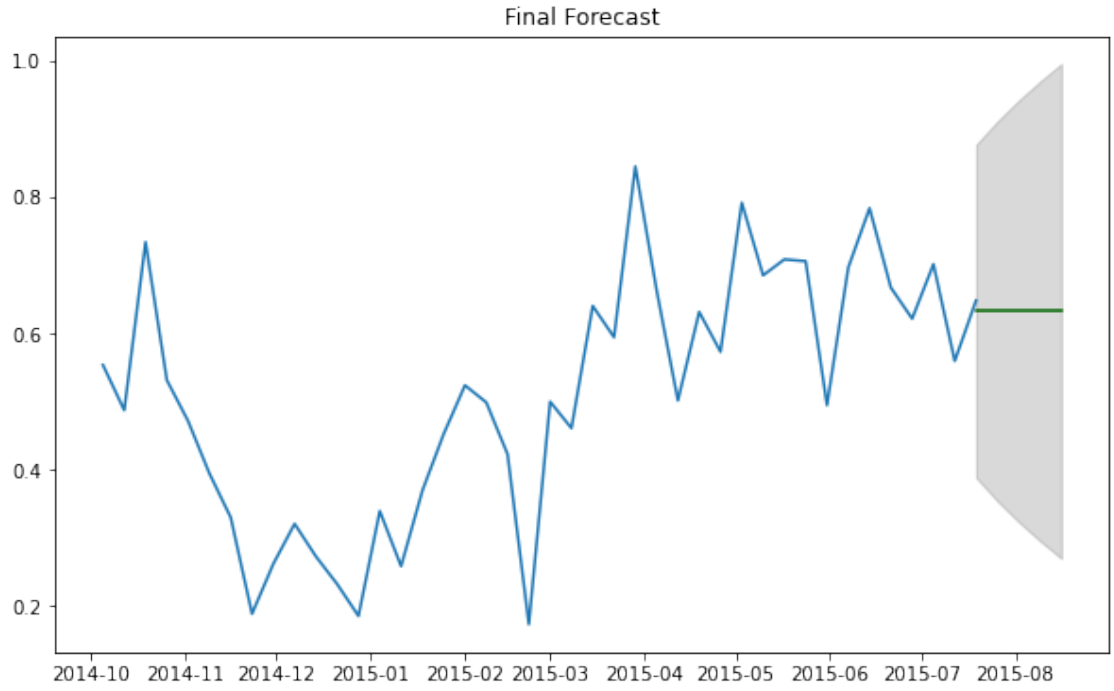
```
C:\Users\scrmo\anaconda3\lib\site-packages\statsmodels\graphics\gofplots.py:993:
UserWarning: marker is redundantly defined by the 'marker' keyword argument and
the fmt string "bo" (-> marker='o'). The keyword argument will take precedence.
  ax.plot(x, y, fmt, **plot_style)
```



Plot 1: Residual errors seem to fluctuate around a mean of zero and have a uniform variance.
Plot 2: The density plot suggest normal distribution with mean zero.
Plot 3: QQ plot seems fine.
Plot 4: ACF plot shows the residual errors are not autocorrelated.

7 Forecast

[38]:
```
# Forecast
n_periods = 5
fc, confint = model.predict(n_periods=n_periods, return_conf_int=True)
index_of_fc = pd.date_range(df.index[-1], periods = n_periods, freq='W')

# make series for plotting purpose
fc_series = pd.Series(fc, index=index_of_fc)
lower_series = pd.Series(confint[:, 0], index=index_of_fc)
upper_series = pd.Series(confint[:, 1], index=index_of_fc)

# Plot
plt.figure(figsize = (10,6))
plt.plot(df.kW_Gen)
plt.plot(fc_series, color='darkgreen')
plt.fill_between(lower_series.index,
                 lower_series,
                 upper_series,
                 color='k', alpha=.15)

plt.title("Final Forecast")
plt.show()
```



8 Cross Validation

Creating training and testing split

[39]:
```
test_size = 24

df_train = df[:-test_size]
df_test = df[-test_size:]
```

[41]:
```
model = pm.auto_arima(df_train.kW_Gen, start_p=1, start_q=1,
                      test='adf',       # use adftest to find optimal 'd'
                      max_p=3, max_q=3, # maximum p and q
                      m=1,              # frequency of series
                      d=None,           # let model determine 'd'
                      seasonal=False,   # No Seasonality
                      start_P=0,
                      D=0,
                      trace=True,
                      error_action='ignore',
                      suppress_warnings=True,
                      stepwise=True)

print(model.summary())
```

```
Performing stepwise search to minimize aic
 ARIMA(1,2,1)(0,0,0)[0] intercept   : AIC=inf, Time=0.10 sec
 ARIMA(0,2,0)(0,0,0)[0] intercept   : AIC=-5.543, Time=0.01 sec
 ARIMA(1,2,0)(0,0,0)[0] intercept   : AIC=-13.414, Time=0.02 sec
 ARIMA(0,2,1)(0,0,0)[0] intercept   : AIC=-16.664, Time=0.04 sec
 ARIMA(0,2,0)(0,0,0)[0]             : AIC=-7.506, Time=0.03 sec
 ARIMA(0,2,2)(0,0,0)[0] intercept   : AIC=-16.368, Time=0.12 sec
 ARIMA(1,2,2)(0,0,0)[0] intercept   : AIC=-15.267, Time=0.13 sec
 ARIMA(0,2,1)(0,0,0)[0]             : AIC=-17.194, Time=0.02 sec
 ARIMA(1,2,1)(0,0,0)[0]             : AIC=-16.519, Time=0.05 sec
 ARIMA(0,2,2)(0,0,0)[0]             : AIC=-16.524, Time=0.03 sec
 ARIMA(1,2,0)(0,0,0)[0]             : AIC=-15.412, Time=0.02 sec
 ARIMA(1,2,2)(0,0,0)[0]             : AIC=-14.536, Time=0.07 sec

Best model:  ARIMA(0,2,1)(0,0,0)[0]
Total fit time: 0.653 seconds
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                   18
Model:               SARIMAX(0, 2, 1)   Log Likelihood                10.597
Date:                Mon, 15 Nov 2021   AIC                          -17.194
Time:                        16:05:17   BIC                          -15.649
Sample:                             0   HQIC                         -17.115
                                  - 18
```

```
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ma.L1         -0.9016      0.561     -1.606      0.108      -2.002       0.199
sigma2         0.0140      0.009      1.554      0.120      -0.004       0.032
==============================================================================
===
Ljung-Box (L1) (Q):                   1.06   Jarque-Bera (JB):
0.14
Prob(Q):                              0.30   Prob(JB):
0.93
Heteroskedasticity (H):               0.54   Skew:
-0.23
Prob(H) (two-sided):                  0.52   Kurtosis:
2.94
==============================================================================
===

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

Predicting the `kW_Gen` using the `auto_arima` model.

[42]:
```
df_test['ARIMA'] = model.predict(len(df_test))

df_test.head()
```

```
C:\Users\scrmo\AppData\Local\Temp/ipykernel_20060/2579051965.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_test['ARIMA'] = model.predict(len(df_test))
```

[42]:

EDT	kW_Gen	ARIMA
2015-02-08	0.498559	0.535945
2015-02-15	0.422483	0.548363
2015-02-22	0.173000	0.560781
2015-03-01	0.499498	0.573199
2015-03-08	0.460746	0.585616

`MAPE` - Mean Absolute Percentage Error

[44]:
```
mean_absolute_percentage_error(df_test.kW_Gen, df_test.ARIMA)
```

[44]: 0.249146925790151