Week10_Python_Demo_BAN5753.ipynb

November 8, 2021

[1]:
```
import numpy as np
import pandas as pd
import saspy
import matplotlib.pyplot as plt
import matplotlib.dates as mdates


# For creating lag scatter plots
from pandas.plotting import lag_plot


from statsmodels.tsa.api import ExponentialSmoothing, SimpleExpSmoothing, Holt


%matplotlib inline
plt.style.use('tableau-colorblind10')
```

1 Loading Data

[2]:
```
df = pd.read_sas("ecommerce.sas7bdat")
```

2 Data Description

The timestep in the data is quarterly.
The `date` column is in `yyyy-mm-dd` format.

[3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 57 entries, 0 to 56
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   date       57 non-null     datetime64[ns]
 1   Ecommerce  57 non-null     float64
dtypes: datetime64[ns](1), float64(1)
memory usage: 1.0 KB
```

[4]:
```
# First row

df.iloc[0]
```

[4]:
```
date         1999-10-01 00:00:00
Ecommerce                 5284.0
Name: 0, dtype: object
```

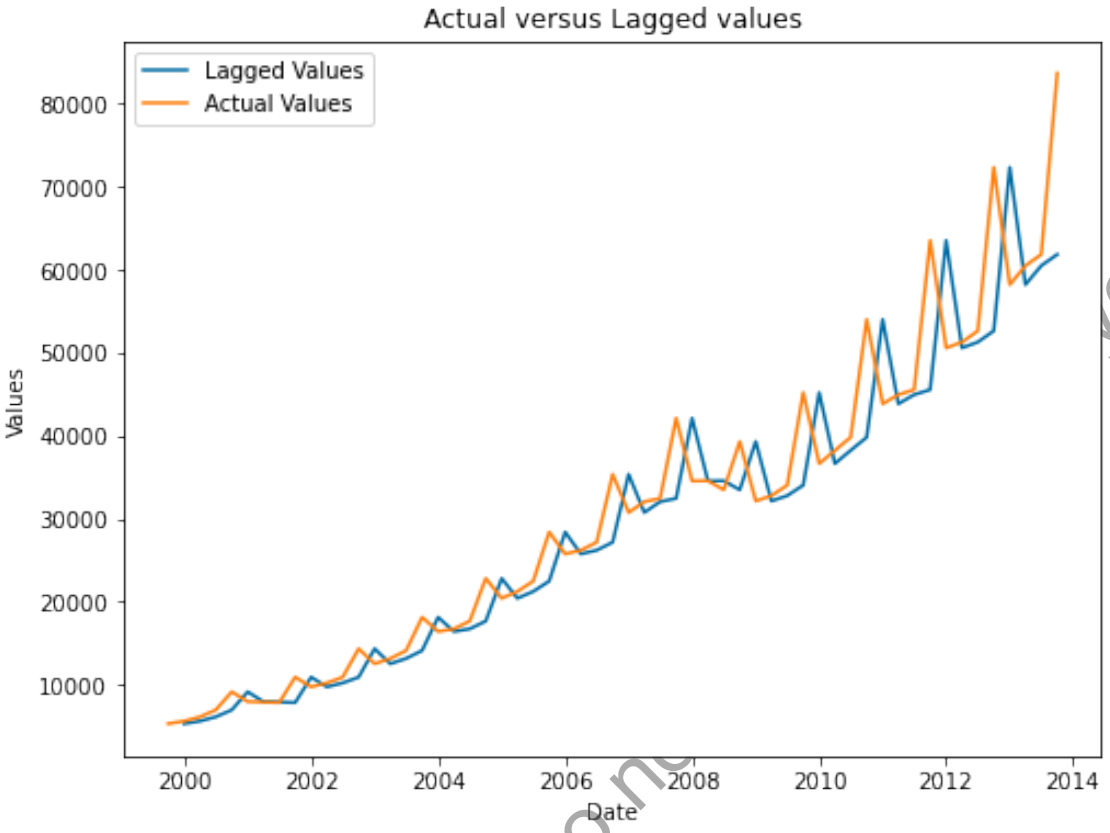
[5]:
```
# Last row

df.iloc[-1]
```

[5]:
```
date         2013-10-01 00:00:00
Ecommerce                83709.0
Name: 56, dtype: object
```

3 Naive ForecastQS-OCT

Plotting relationship between each observation and a lag of that observation.

[6]:
```
# Ceating a dummy dataframe containing a lag column of the data
df1 = df.copy()
df1['Ecommerce_lag'] = df1['Ecommerce'].shift(1)


plt.figure(figsize=(8,6))
plt.plot(df1['date'],df1['Ecommerce_lag'])
plt.plot(df1['date'],df1['Ecommerce'])
plt.title('Actual versus Lagged values')
plt.xlabel('Date')
plt.ylabel('Values')
plt.legend(['Lagged Values', 'Actual Values'])
plt.show()
```



4 Data Preparation

Create a `pandas Series` with the index being the `date`.

`QS-OCT` means Quarter Starting in October.

[7]:
```
index = pd.date_range(start="1999-Q1", end="2013-Q1", freq="QS-OCT")
data = pd.Series(df['Ecommerce'].to_list(), index)

data.head()
```

[7]:
```
1999-01-01    5284.0
1999-04-01    5590.0
1999-07-01    6100.0
1999-10-01    6936.0
2000-01-01    9126.0
Freq: QS-OCT, dtype: float64
```

5 ESM Model with no trend or seasonality

You can put explicit value of the smoothing parameter α by specifying the `smoothing_level` in the `fit()` method.

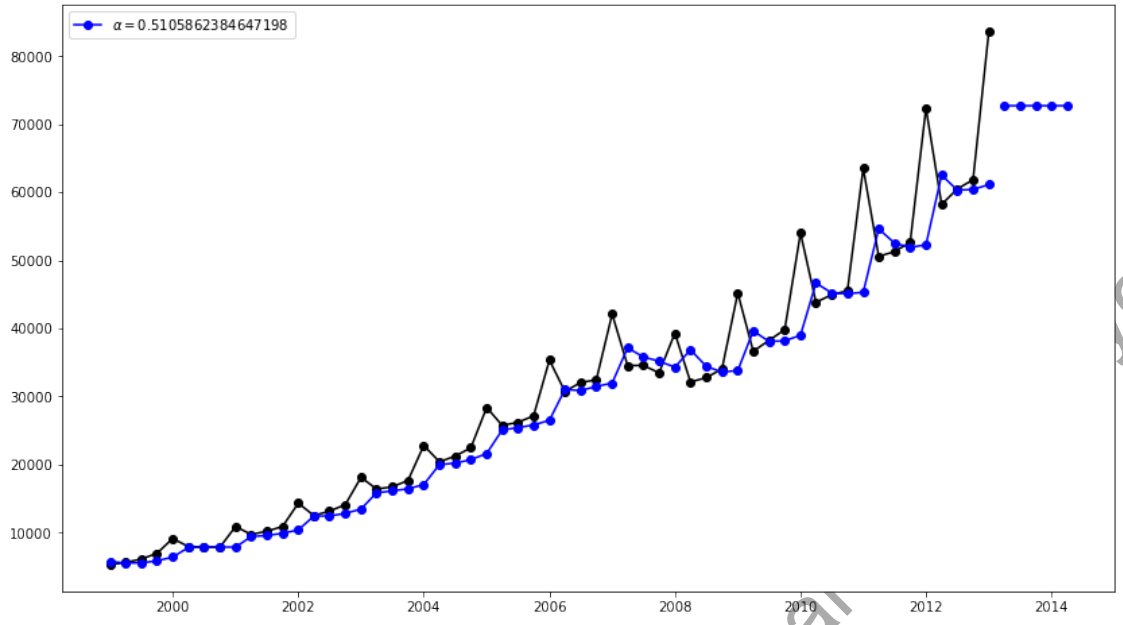
The model here will find the optimized α

Forecast is for 5 steps forward

[8]:
```
forecast_steps = 5

# Model fitting and forecasting

# alpha is auto-optimized
fit1 = SimpleExpSmoothing(data, initialization_method="estimated").fit()
fcast1 = fit1.forecast(forecast_steps).rename(r"$\alpha=%s$" % fit1.model.
 ↪params["smoothing_level"])


# Forecast Plotting

plt.figure(figsize=(14, 8))
plt.plot(data, marker="o", color="black")
plt.plot(fit1.fittedvalues, marker="o", color="blue")
(line,) = plt.plot(fcast1, marker="o", color="blue")
plt.legend([line], [fcast1.name])
```

```
C:\Users\scrmo\anaconda3\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:132: FutureWarning: The 'freq'
argument in Timestamp is deprecated and will be removed in a future version.
  date_key = Timestamp(key, freq=base_index.freq)
```

[8]:
```
<matplotlib.legend.Legend at 0x25d8961f160>
```



6 ESM Model with linear trend but no seasonality

You can put explicit value of the parameters α and β by specifying the `smoothing_level` and `smoothing_trend` in the `fit()` method.

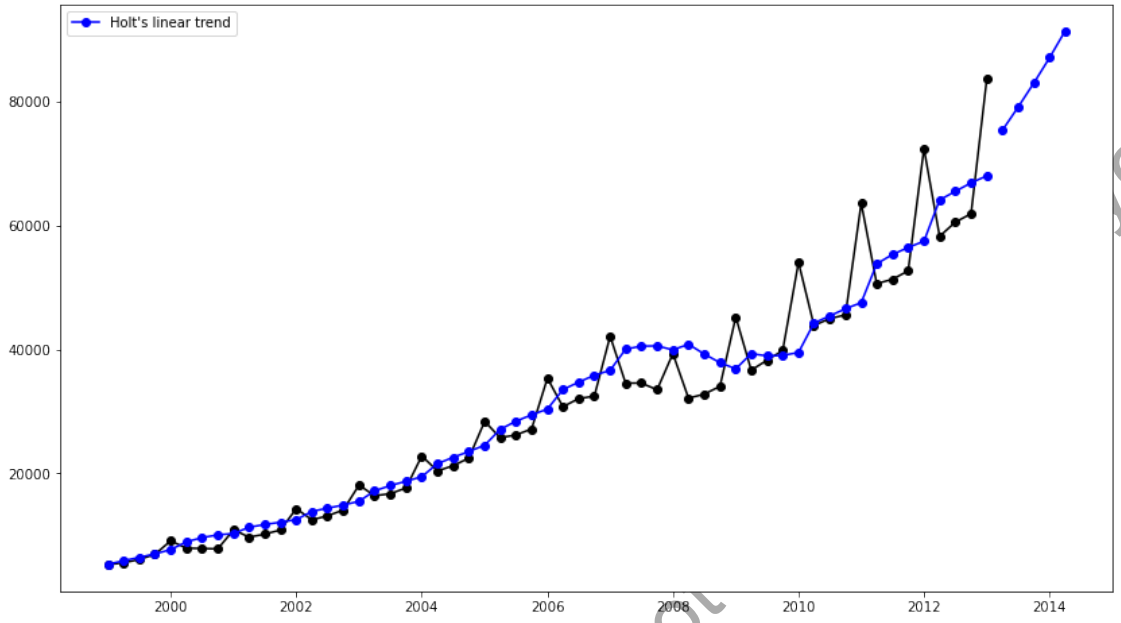
Insert parameter `exponential=True` in the method `Holt` for linear trend.

[9]:
```
forecast_steps = 5

# Model fitting and forecasting

# alpha and beta is auto-optimized
fit2 = Holt(data, exponential=True, initialization_method="estimated").fit()
fcast2 = fit2.forecast(forecast_steps).rename("Holt's linear trend")


# Forecast Plotting

plt.figure(figsize=(14, 8))
plt.plot(data, marker="o", color="black")
plt.plot(fit2.fittedvalues, marker="o", color="blue")
(line,) = plt.plot(fcast2, marker="o", color="blue")
plt.legend([line], [fcast2.name])
```

```
C:\Users\scrmo\anaconda3\lib\site-
packages\statsmodels\tsa\holtwinters\model.py:920: ConvergenceWarning:
Optimization failed to converge. Check mle_retvals.
```

```
  warnings.warn(
```

[9]:
```
<matplotlib.legend.Legend at 0x25d89badf70>
```



7 ESM Model with linear trend and additive seasonality

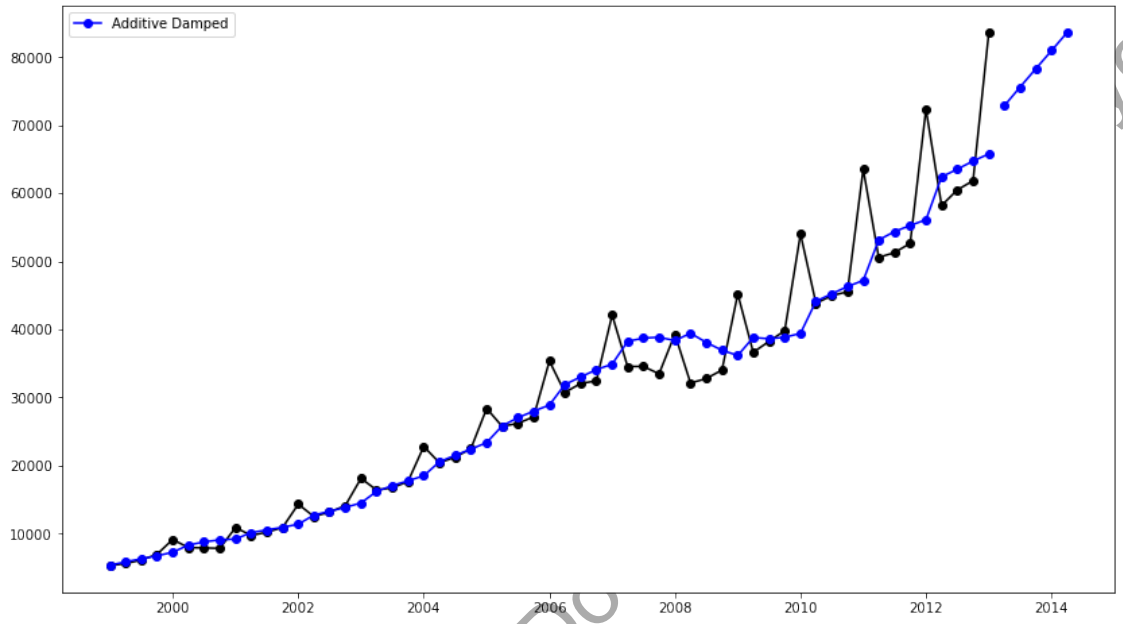
You can put explicit value of the parameters α, β and ϕ by specifying the `smoothing_level`, `smoothing_trend` and `damped_trend` in the `fit()` method.

Insert parameter `damped_trend=True` in the method `Holt` for additive sesonality.

[10]:
```
forecast_steps = 5

# Model fitting and forecasting

# alpha, beta and phi is auto-optimized
fit3 = Holt(data, damped_trend=True, initialization_method="estimated").fit()
fcast3 = fit3.forecast(forecast_steps).rename("Additive Damped")


# Forecast Plotting

plt.figure(figsize=(14, 8))
plt.plot(data, marker="o", color="black")
plt.plot(fit3.fittedvalues, marker="o", color="blue")
(line,) = plt.plot(fcast3, marker="o", color="blue")
plt.legend([line], [fcast3.name])
```

```
C:\Users\scrmo\anaconda3\lib\site-
packages\statsmodels\tsa\holtwinters\model.py:920: ConvergenceWarning:
Optimization failed to converge. Check mle_retvals.
  warnings.warn(
```

[10]: `<matplotlib.legend.Legend at 0x25d89e2fc40>`



8 ESM Model with linear trend and multiplicative seasonality

You can put explicit value of the parameters , and by specifying the smoothing_level, smoothing_trend and damped_trend in the fit() method.

Insert parameter `damped_trend=True` and `exponential=True` in the method `Holt` for multiplicative sesonality.

[11]:
```
forecast_steps = 5

# Model fitting and forecasting

# alpha, beta and phi is auto-optimized
fit4 = Holt(data, exponential=True, damped_trend=True,␣
 ↪initialization_method="estimated").fit()
fcast4 = fit4.forecast(forecast_steps).rename("Multiplicative Damped")


# Forecast Plotting
```

```
plt.figure(figsize=(14, 8))
plt.plot(data, marker="o", color="black")
plt.plot(fit4.fittedvalues, marker="o", color="blue")
(line,) = plt.plot(fcast4, marker="o", color="blue")
plt.legend([line], [fcast4.name])
```

```
C:\Users\scrmo\anaconda3\lib\site-
packages\statsmodels\tsa\holtwinters\model.py:920: ConvergenceWarning:
Optimization failed to converge. Check mle_retvals.
  warnings.warn(
```

[11]: <matplotlib.legend.Legend at 0x25d89c56fa0>