American_prophet

December 19, 2020

[1]:
```
from fbprophet import Prophet
import pandas as pd
from pandas import DataFrame
from matplotlib import pyplot
#univariate
```

[2]:
```
turkey = pd.read_excel("ecomretailfixed.xls",header=0)
```

[3]:
```
turkey.columns = ['ds', 'y']
turkey
turkey['ds']= pd.to_datetime(turkey['ds'])
```

[4]:
```
model = Prophet()
```

[5]:
```
model.fit(turkey)
```

```
INFO:numexpr.utils:NumExpr defaulting to 8 threads.
INFO:fbprophet:Disabling weekly seasonality. Run prophet with
weekly_seasonality=True to override this.
INFO:fbprophet:Disabling daily seasonality. Run prophet with
daily_seasonality=True to override this.
```

[5]:
```
<fbprophet.forecaster.Prophet at 0x7f9e4470a0a0>
```

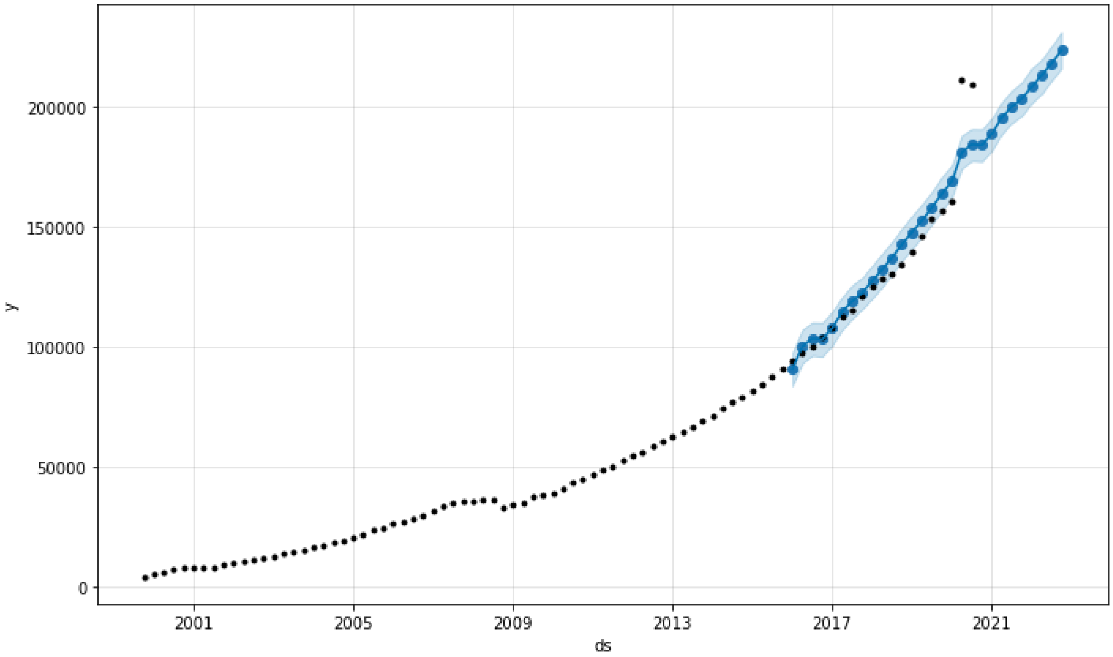
[6]:
```
# define the period for which we want a prediction
future = list()
while True:
    date_in = input("enter year or enter (q)uit" )
    if date_in =="q":
        break
    for i in range(1,12,3):
        date = "{0}-{1}".format(date_in,i)
        future.append([date])
    print(future)
print(future)
future = DataFrame(future)
future.columns = ['ds']
future['ds']= pd.to_datetime(future['ds'])
```

```python
# use the model to make a forecast
forecast = model.predict(future)
# summarize the forecast
print(forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']])
# plot forecast

model.plot(forecast)
pyplot.scatter(forecast['ds'],forecast['yhat'])
pyplot.show()
```

```
enter year or enter (q)uit2016
[['2016-1'], ['2016-4'], ['2016-7'], ['2016-10']]
enter year or enter (q)uit2017
[['2016-1'], ['2016-4'], ['2016-7'], ['2016-10'], ['2017-1'], ['2017-4'],
['2017-7'], ['2017-10']]
enter year or enter (q)uit2018
[['2016-1'], ['2016-4'], ['2016-7'], ['2016-10'], ['2017-1'], ['2017-4'],
['2017-7'], ['2017-10'], ['2018-1'], ['2018-4'], ['2018-7'], ['2018-10']]
enter year or enter (q)uit2019
[['2016-1'], ['2016-4'], ['2016-7'], ['2016-10'], ['2017-1'], ['2017-4'],
['2017-7'], ['2017-10'], ['2018-1'], ['2018-4'], ['2018-7'], ['2018-10'],
['2019-1'], ['2019-4'], ['2019-7'], ['2019-10']]
enter year or enter (q)uit2020
[['2016-1'], ['2016-4'], ['2016-7'], ['2016-10'], ['2017-1'], ['2017-4'],
['2017-7'], ['2017-10'], ['2018-1'], ['2018-4'], ['2018-7'], ['2018-10'],
['2019-1'], ['2019-4'], ['2019-7'], ['2019-10'], ['2020-1'], ['2020-4'],
['2020-7'], ['2020-10']]
enter year or enter (q)uit2021
[['2016-1'], ['2016-4'], ['2016-7'], ['2016-10'], ['2017-1'], ['2017-4'],
['2017-7'], ['2017-10'], ['2018-1'], ['2018-4'], ['2018-7'], ['2018-10'],
['2019-1'], ['2019-4'], ['2019-7'], ['2019-10'], ['2020-1'], ['2020-4'],
['2020-7'], ['2020-10'], ['2021-1'], ['2021-4'], ['2021-7'], ['2021-10']]
enter year or enter (q)uit2022
[['2016-1'], ['2016-4'], ['2016-7'], ['2016-10'], ['2017-1'], ['2017-4'],
['2017-7'], ['2017-10'], ['2018-1'], ['2018-4'], ['2018-7'], ['2018-10'],
['2019-1'], ['2019-4'], ['2019-7'], ['2019-10'], ['2020-1'], ['2020-4'],
['2020-7'], ['2020-10'], ['2021-1'], ['2021-4'], ['2021-7'], ['2021-10'],
['2022-1'], ['2022-4'], ['2022-7'], ['2022-10']]
enter year or enter (q)uitq
[['2016-1'], ['2016-4'], ['2016-7'], ['2016-10'], ['2017-1'], ['2017-4'],
['2017-7'], ['2017-10'], ['2018-1'], ['2018-4'], ['2018-7'], ['2018-10'],
['2019-1'], ['2019-4'], ['2019-7'], ['2019-10'], ['2020-1'], ['2020-4'],
['2020-7'], ['2020-10'], ['2021-1'], ['2021-4'], ['2021-7'], ['2021-10'],
['2022-1'], ['2022-4'], ['2022-7'], ['2022-10']]
           ds           yhat     yhat_lower     yhat_upper
0  2016-01-01   90779.234761   83782.322881   97848.267277
1  2016-04-01  100197.314307   93256.254192  107342.128530
```

```
2  2016-07-01  103568.387154   96502.248579  110397.931410
3  2016-10-01  103242.524072   96113.495603  110252.915959
4  2017-01-01  107927.048444  100795.798696  114934.646466
5  2017-04-01  114776.610909  107179.868689  120986.600259
6  2017-07-01  119098.537540  111797.309863  125795.798983
7  2017-10-01  122546.882308  115759.607073  129015.367709
8  2018-01-01  127523.177319  120258.194744  134045.854275
9  2018-04-01  132231.198535  124916.745851  139054.296917
10 2018-07-01  137247.191358  130258.089583  143959.661097
11 2018-10-01  142753.402969  135999.042302  149598.608416
12 2019-01-01  147896.585391  141013.809708  154807.342333
13 2019-04-01  152550.884266  145747.528215  159513.975559
14 2019-07-01  158029.005843  151136.943457  164836.779537
15 2019-10-01  163874.013897  156757.225999  171062.641694
16 2020-01-01  169034.596032  162499.018540  175828.203035
17 2020-04-01  181098.080732  174419.963429  188326.110529
18 2020-07-01  184469.153579  177598.130567  190905.105474
19 2020-10-01  184143.290497  177302.999745  190815.270135
20 2021-01-01  188827.814869  181904.872538  195481.872390
21 2021-04-01  195677.377334  188741.191221  202155.484597
22 2021-07-01  199999.303966  193297.030967  206813.919174
23 2021-10-01  203447.648733  196322.285492  210432.467003
24 2022-01-01  208423.943743  201707.304283  216293.735591
25 2022-04-01  213131.964960  205430.126536  219840.080505
26 2022-07-01  218147.957783  211077.781616  225504.000305
27 2022-10-01  223654.169394  216046.020774  231340.690869
```

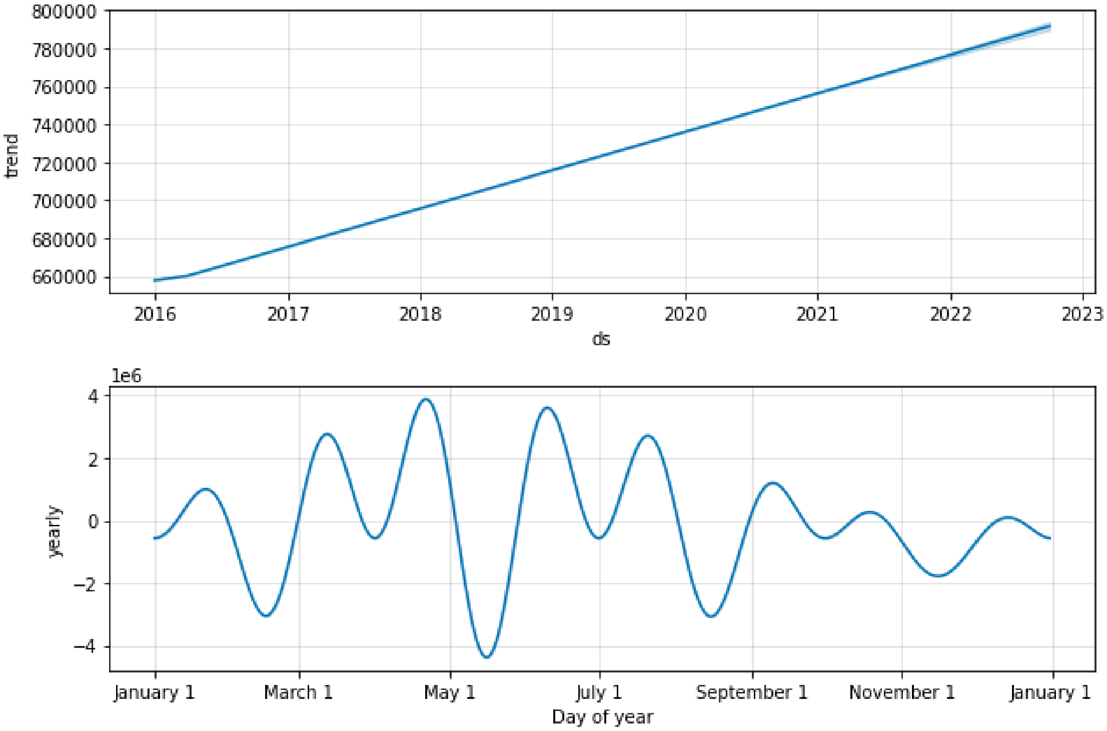


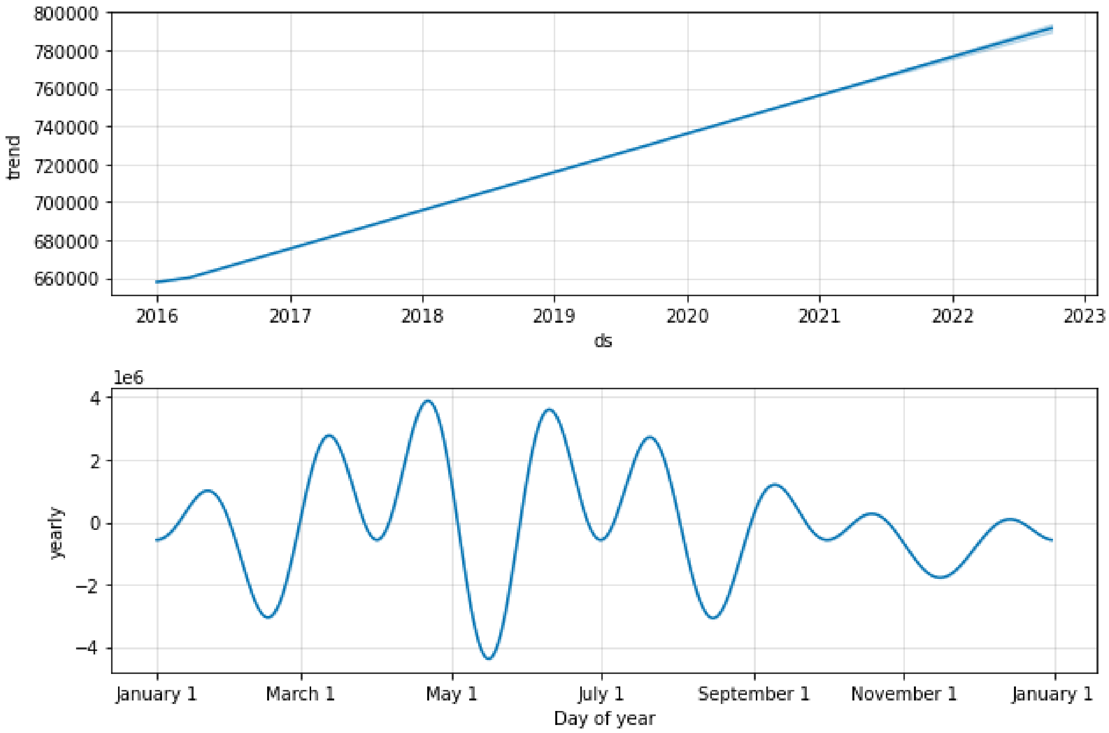
[7]:

```
# create test dataset, in quarters length
train = turkey.drop(turkey.index[:-18])
print(train)
model.plot_components(forecast)
```

```
            ds       y
66  2016-04-01   97459
67  2016-07-01  100519
68  2016-10-01  103952
69  2017-01-01  108157
70  2017-04-01  112644
71  2017-07-01  115419
72  2017-10-01  121019
73  2018-01-01  124936
74  2018-04-01  128616
75  2018-07-01  130625
76  2018-10-01  134291
77  2019-01-01  139713
78  2019-04-01  146394
79  2019-07-01  153224
80  2019-10-01  156581
81  2020-01-01  160414
82  2020-04-01  211595
83  2020-07-01  209533
```

[7]:





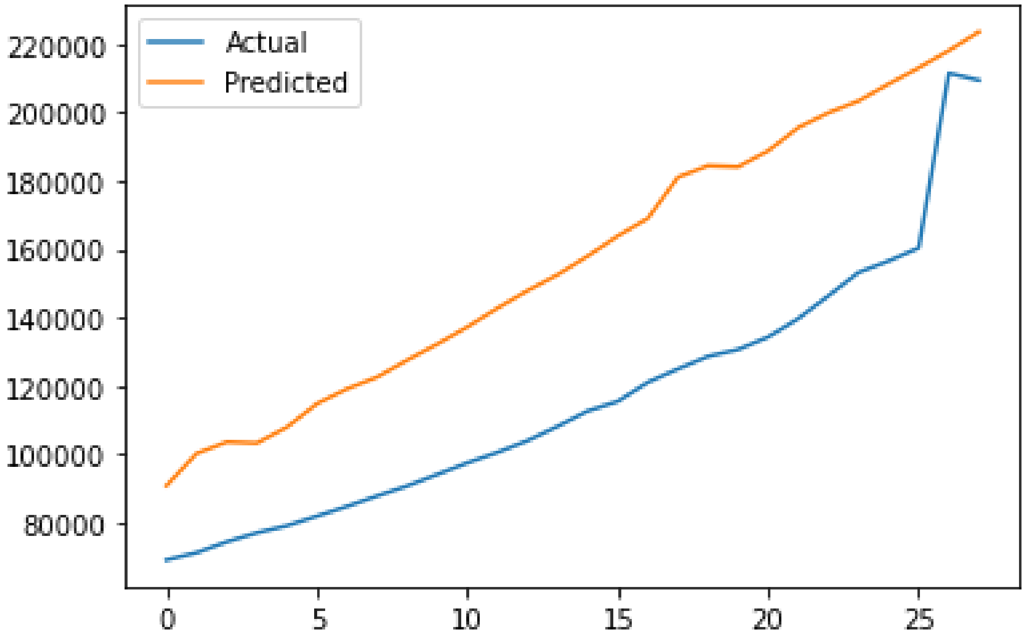
```
[8]: len(forecast['yhat'].values)
```

```
[8]: 28
```

```
[9]: from sklearn.metrics import mean_absolute_error, mean_squared_log_error,␣
      ↪balanced_accuracy_score
     # calculate MAE between expected and predicted values for december
     y_true = turkey['y'][-len(future):].values
     y_pred = forecast['yhat'].values
     mae = mean_absolute_error(y_true, y_pred)
     loss = mean_squared_log_error(y_true,y_pred)
     print("loss score",loss)
     print('MAE: %.3f' % mae)
```

```
loss score 0.101927960631951
MAE: 40349.904
```

```
[10]: # plot expected vs actual
      pyplot.plot(y_true, label='Actual')
      pyplot.plot(y_pred, label='Predicted')
      pyplot.legend()
      pyplot.show()
```



[11]:
```
from fbprophet.plot import plot_plotly, plot_components_plotly

plot_plotly(model, forecast)
```

[12]:
```
plot_components_plotly(model, forecast)
```

[ ]: