Prophet_Turkey

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("TurkeyData3.xlsx",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 0x7f7ba592d040>
```

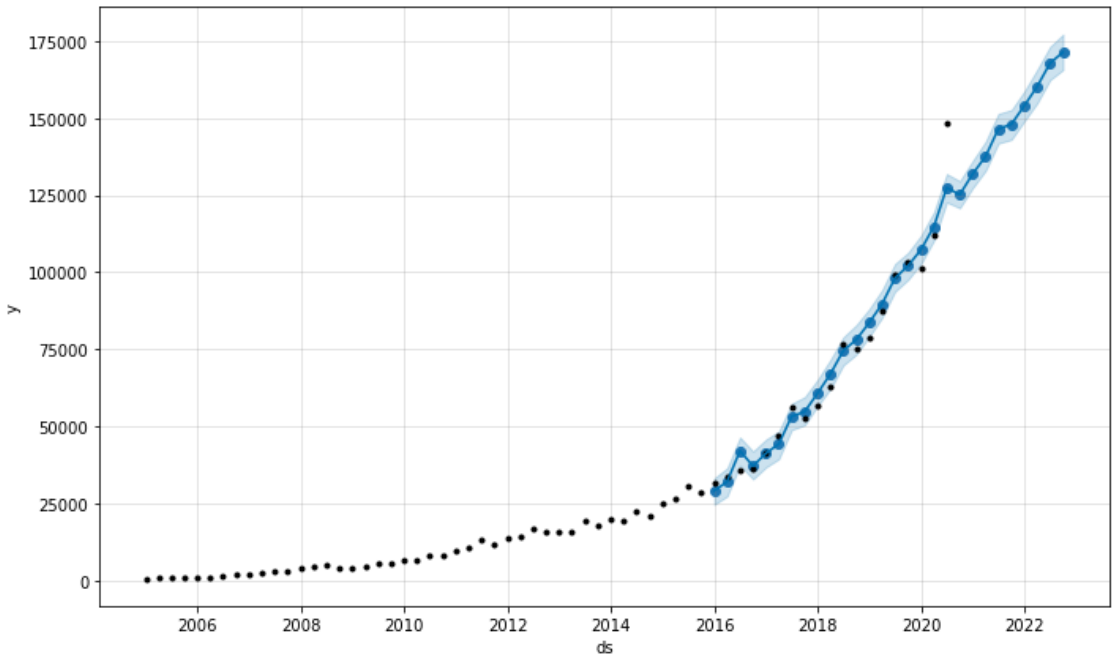
[8]:
```
# define the period for which we want a prediction
future = list()
while True:
    date_in = input("enter year or enter (q)uit" )
    if date_in =="q":
        break
    for i in range(1,12,3):
        date = "{0}-{1}".format(date_in,i)
        future.append([date])
    print(future)
print(future)
future = DataFrame(future)
future.columns = ['ds']
future['ds']= pd.to_datetime(future['ds'])
```

```
# 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   29094.201917   24631.786554   33200.107170
1  2016-04-01   32012.680777   27302.972218   36471.837735
```

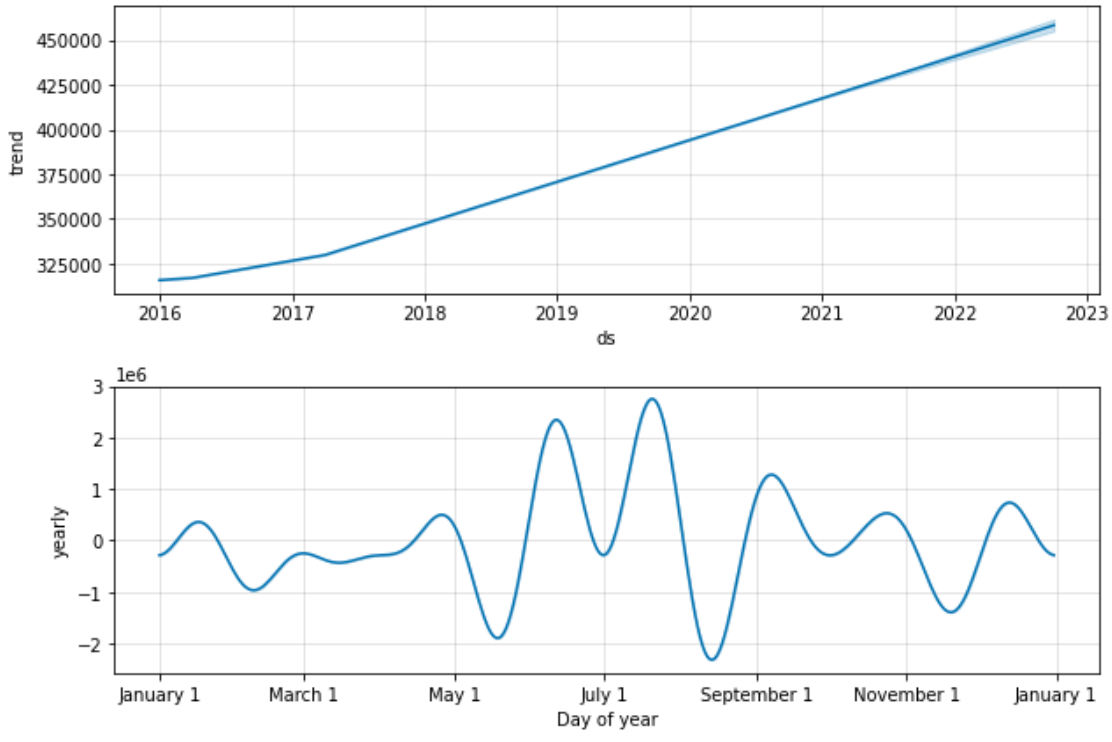
2	2016-07-01	42060.280124	37367.024873	46448.066932
3	2016-10-01	37131.392173	32849.480596	41890.964426
4	2017-01-01	41264.973385	36755.216281	45784.204848
5	2017-04-01	44242.266662	39396.481793	48414.538258
6	2017-07-01	53229.171147	48873.194268	57395.420438
7	2017-10-01	54748.735439	50398.155548	59554.654438
8	2018-01-01	60785.826399	56634.696695	65232.119511
9	2018-04-01	66989.322771	62153.530722	71680.753023
10	2018-07-01	74557.706923	69858.437065	78988.453253
11	2018-10-01	78147.965021	73355.185437	82962.189069
12	2019-01-01	83676.019674	79161.399925	88199.395537
13	2019-04-01	89706.300298	85289.070519	94298.362585
14	2019-07-01	98133.630061	93762.231948	102775.716851
15	2019-10-01	102084.309275	97587.024970	106399.343629
16	2020-01-01	107344.826386	102771.198184	111964.057702
17	2020-04-01	114801.011325	110325.818235	119539.441431
18	2020-07-01	127476.917156	122766.565151	131983.246890
19	2020-10-01	125205.218178	120866.167723	129646.122196
20	2021-01-01	131994.082392	127447.469167	136296.093554
21	2021-04-01	137568.935126	132983.625469	142449.105358
22	2021-07-01	146555.839611	141874.111232	151514.390977
23	2021-10-01	148075.403904	143096.045427	152733.800414
24	2022-01-01	154112.494863	149137.624984	158865.577375
25	2022-04-01	160315.991236	154922.222399	165832.985071
26	2022-07-01	167884.375388	162470.895534	173282.320846
27	2022-10-01	171474.633485	165769.209471	177270.006747

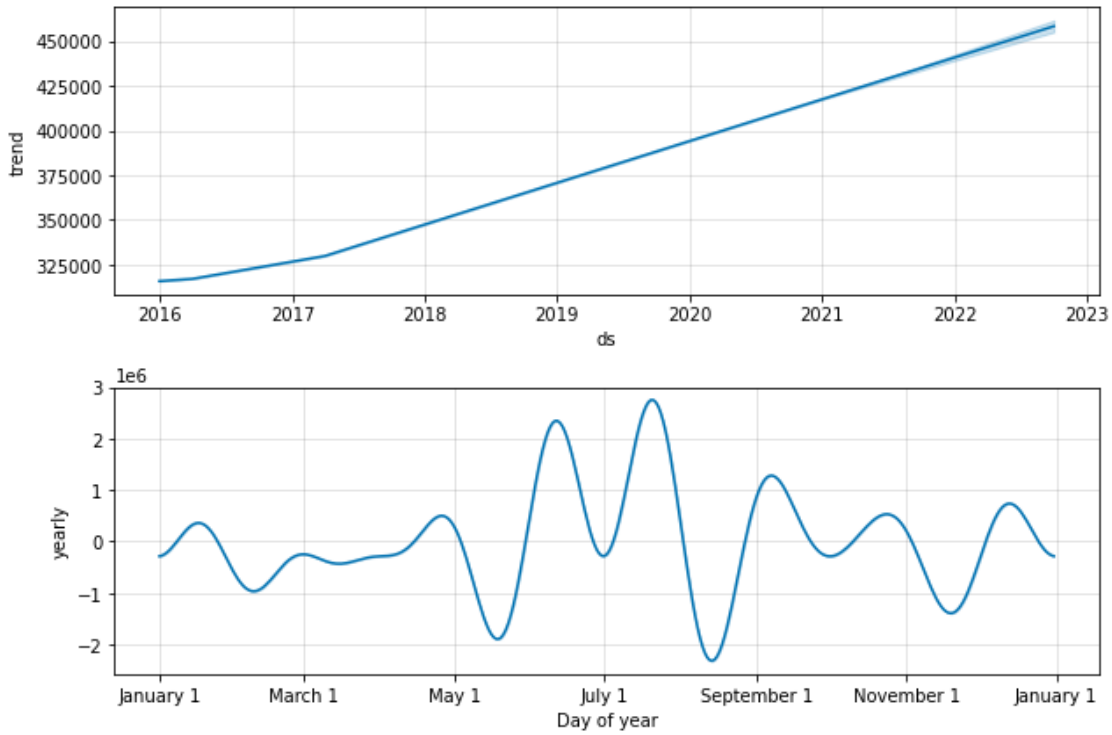


[9]:
```
# create test dataset, in quarters length
train = turkey.drop(turkey.index[:-18])
print(train)
model.plot_components(forecast)
```

```
           ds          y
45 2016-04-01   33540.70
46 2016-07-01   35564.10
47 2016-10-01   36377.03
48 2017-01-01   41252.10
49 2017-04-01   47138.21
50 2017-07-01   56096.90
51 2017-10-01   52838.28
52 2018-01-01   56672.95
53 2018-04-01   63083.56
54 2018-07-01   76672.68
55 2018-10-01   75088.54
56 2019-01-01   78769.81
57 2019-04-01   87255.83
58 2019-07-01   99103.90
59 2019-10-01  103265.11
60 2020-01-01  101339.37
61 2020-04-01  111949.08
62 2020-07-01  148634.52
```

[9]:





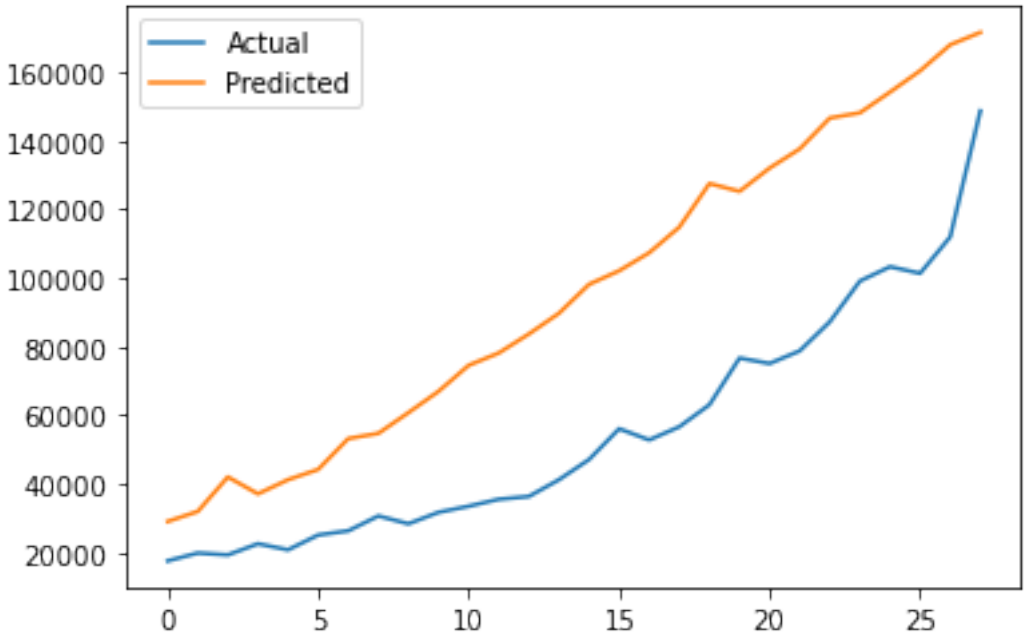
[10]:
```
len(forecast['yhat'].values)
```

[10]: 28

[11]:
```
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 0.38828359395967255
MAE: 40472.427
```

[12]:
```
# plot expected vs actual
pyplot.plot(y_true, label='Actual')
pyplot.plot(y_pred, label='Predicted')
pyplot.legend()
pyplot.show()
```



[13]:
```
from fbprophet.plot import plot_plotly, plot_components_plotly

plot_plotly(model, forecast)
```

[14]:
```
plot_components_plotly(model, forecast)
```

[ ]:
```

```