00-Intro-to-Facebook-Prophet

October 19, 2022

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

0.1 # Quick Guide to Facebook's Prophet Basics

0.2 IMPORTANT NOTE ONE:

You should really read the papaer for Prophet! It is relatively straightforward and has a lot of insight on their techniques on how Prophet works internally!

0.3 Link to paper: https://peerj.com/preprints/3190.pdf

0.4 IMPORTANT NOTE TWO:

0.5 -----

- **NOTE: Link to installation instructions:**
  - https://facebook.github.io/prophet/docs/installation.html#python
  - SCROLL DOWN UNTIL YOU SEE THE ANACONDA OPTION AT THE BOTTOM OF THE PAGE.
  - YOU MAY NEED TO INSTALL BOTH **conda install gcc** and **conda install -c conda-forge fbprophet**
  - PLEASE READ THROUGH THE DOCS AND STACKOERFLOW CAREFULLY BEFORE POSTING INSTALLATION ISSUES TO THE QA FORUMS.

0.6 -----

0.7 Load Libraries

[1]:
```
import pandas as pd
from fbprophet import Prophet
```

0.8 Load Data

The input to Prophet is always a dataframe with two columns: ds and y. The ds (datestamp) column should be of a format expected by Pandas, ideally YYYY-MM-DD for a date or YYYY-MM-DD HH:MM:SS for a timestamp. The y column must be numeric, and represents the measurement we wish to forecast.

[2]:
```
df = pd.read_csv('../Data/BeerWineLiquor.csv')
```

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

[3]:

	date	beer
0	1/1/1992	1509
1	2/1/1992	1541
2	3/1/1992	1597
3	4/1/1992	1675
4	5/1/1992	1822

0.8.1 Format the Data

[4]:
```
df.columns = ['ds','y']
```

[5]:
```
df['ds'] = pd.to_datetime(df['ds'])
```

0.9 Create and Fit Model

[6]:
```
# This is fitting on all the data (no train test split in this example)
m = Prophet()
m.fit(df)
```

```
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.
/anaconda3/lib/python3.6/site-packages/pystan/misc.py:399: FutureWarning:
Conversion of the second argument of issubdtype from `float` to `np.floating` is
deprecated. In future, it will be treated as `np.float64 ==
```

```
np.dtype(float).type`.
  elif np.issubdtype(np.asarray(v).dtype, float):
```

[6]:
```
<fbprophet.forecaster.Prophet at 0x1a17e5ca90>
```

0.10 Forecasting

0.10.1 Step 1: Create "future" placeholder dataframe

NOTE: Prophet by default is for daily data. You need to pass a frequency for sub-daily or monthly data. Info: https://facebook.github.io/prophet/docs/non-daily_data.html

[7]:
```
future = m.make_future_dataframe(periods=24,freq = 'MS')
```

[8]:
```
df.tail()
```

[8]:

	ds	y
319	2018-08-01	4898
320	2018-09-01	4598
321	2018-10-01	4737
322	2018-11-01	5130
323	2018-12-01	6370

[9]:
```
future.tail()
```

[9]:

	ds
343	2020-08-01
344	2020-09-01
345	2020-10-01
346	2020-11-01
347	2020-12-01

[10]:
```
len(df)
```

[10]:
```
324
```

[11]:
```
len(future)
```

[11]:
```
348
```

0.10.2 Step 2: Predict and fill in the Future

[12]:
```
forecast = m.predict(future)
```

[13]:
```
forecast.head()
```

[13]:

	ds	trend	yhat_lower	yhat_upper	trend_lower	trend_upper
0	1992-01-01	1773.449803	1137.970143	1466.120871	1773.449803	1773.449803
1	1992-02-01	1776.239771	1185.694605	1515.005441	1776.239771	1776.239771
2	1992-03-01	1778.849740	1437.426600	1748.679890	1778.849740	1778.849740
3	1992-04-01	1781.639707	1420.155130	1753.209179	1781.639707	1781.639707
4	1992-05-01	1784.339676	1665.749772	1985.649866	1784.339676	1784.339676

	additive_terms	additive_terms_lower	additive_terms_upper	yearly
0	-461.776706	-461.776706	-461.776706	-461.776706
1	-427.591035	-427.591035	-427.591035	-427.591035
2	-179.181320	-179.181320	-179.181320	-179.181320
3	-196.311603	-196.311603	-196.311603	-196.311603
4	45.026915	45.026915	45.026915	45.026915

	yearly_lower	yearly_upper	multiplicative_terms
0	-461.776706	-461.776706	0.0
1	-427.591035	-427.591035	0.0
2	-179.181320	-179.181320	0.0
3	-196.311603	-196.311603	0.0
4	45.026915	45.026915	0.0

	multiplicative_terms_lower	multiplicative_terms_upper	yhat
0	0.0	0.0	1311.673097
1	0.0	0.0	1348.648735
2	0.0	0.0	1599.668420
3	0.0	0.0	1585.328104
4	0.0	0.0	1829.366590

[14]:
```
forecast.tail()
```

[14]:

	ds	trend	yhat_lower	yhat_upper	trend_lower
343	2020-08-01	5201.010945	5074.524666	5397.039919	5190.005465
344	2020-09-01	5217.085200	4932.976994	5246.885844	5205.065711
345	2020-10-01	5232.640931	5037.574360	5361.483095	5219.696878
346	2020-11-01	5248.715186	5168.423030	5502.051767	5234.716079
347	2020-12-01	5264.270916	6160.467499	6497.079944	5249.311816

	trend_upper	additive_terms	additive_terms_lower	additive_terms_upper
343	5213.185170	40.218725	40.218725	40.218725
344	5230.470140	-132.884136	-132.884136	-132.884136
345	5247.521702	-43.275393	-43.275393	-43.275393
346	5264.968845	86.703091	86.703091	86.703091
347	5281.695680	1071.508751	1071.508751	1071.508751

	yearly	yearly_lower	yearly_upper	multiplicative_terms

```
343    40.218725     40.218725     40.218725                  0.0
344  -132.884136   -132.884136   -132.884136                  0.0
345   -43.275393    -43.275393    -43.275393                  0.0
346    86.703091     86.703091     86.703091                  0.0
347  1071.508751   1071.508751   1071.508751                  0.0

     multiplicative_terms_lower  multiplicative_terms_upper         yhat
343                         0.0                         0.0  5241.229670
344                         0.0                         0.0  5084.201064
345                         0.0                         0.0  5189.365538
346                         0.0                         0.0  5335.418277
347                         0.0                         0.0  6335.779667
```

[15]:
```
forecast.columns
```

[15]:
```
Index(['ds', 'trend', 'yhat_lower', 'yhat_upper', 'trend_lower', 'trend_upper',
       'additive_terms', 'additive_terms_lower', 'additive_terms_upper',
       'yearly', 'yearly_lower', 'yearly_upper', 'multiplicative_terms',
       'multiplicative_terms_lower', 'multiplicative_terms_upper', 'yhat'],
      dtype='object')
```

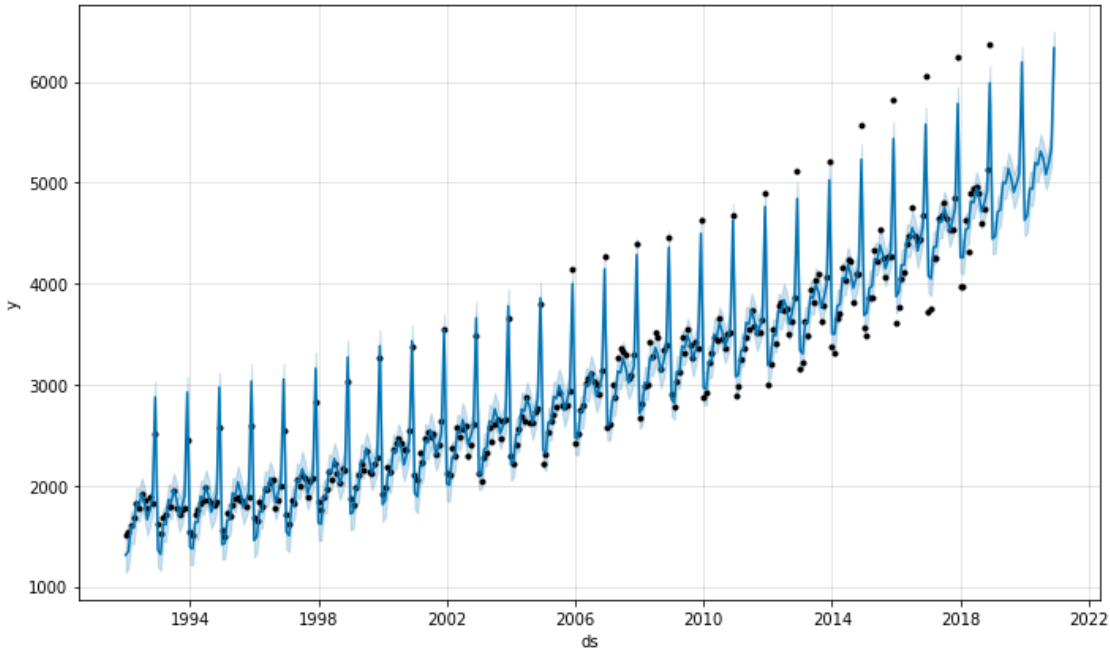
[16]:
```
forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail(12)
```

[16]:
```
           ds         yhat   yhat_lower   yhat_upper
336 2020-01-01  4628.788552  4471.253028  4805.818068
337 2020-02-01  4679.048478  4507.791210  4844.742347
338 2020-03-01  4942.495400  4783.785125  5110.230608
339 2020-04-01  4941.439371  4772.041628  5107.288259
340 2020-05-01  5198.333619  5037.037450  5358.603793
341 2020-06-01  5179.412285  5009.729943  5337.560681
342 2020-07-01  5311.959526  5151.814252  5480.267144
343 2020-08-01  5241.229670  5074.524666  5397.039919
344 2020-09-01  5084.201064  4932.976994  5246.885844
345 2020-10-01  5189.365538  5037.574360  5361.483095
346 2020-11-01  5335.418277  5168.423030  5502.051767
347 2020-12-01  6335.779667  6160.467499  6497.079944
```

0.10.3 Plotting Forecast

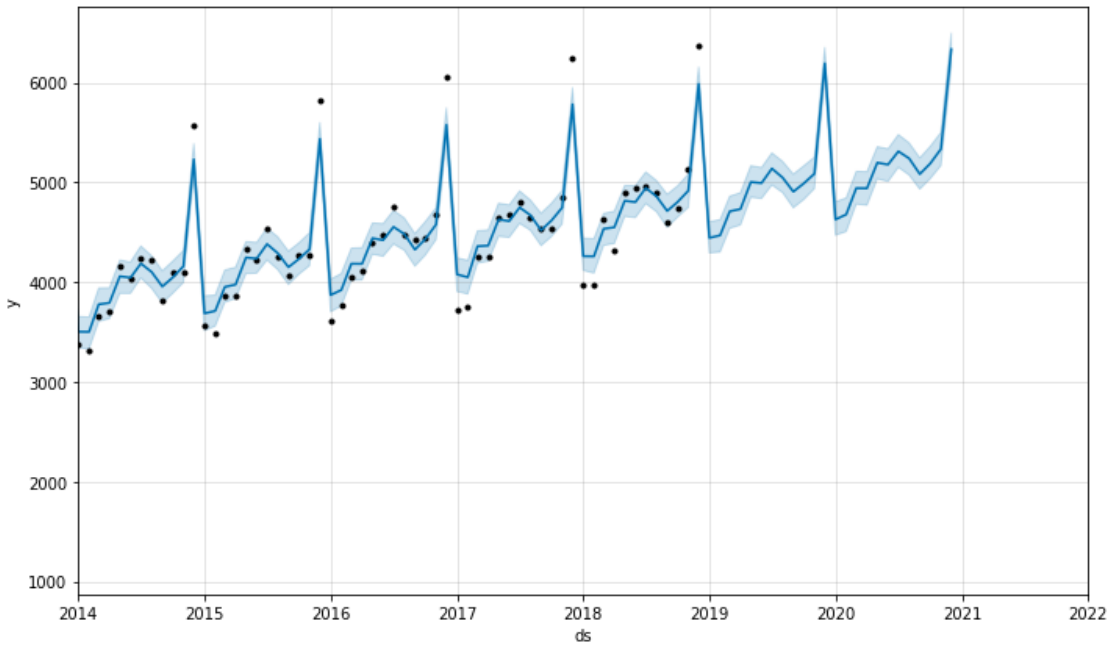
We can use Prophet's own built in plotting tools

[17]:
```
m.plot(forecast);
```



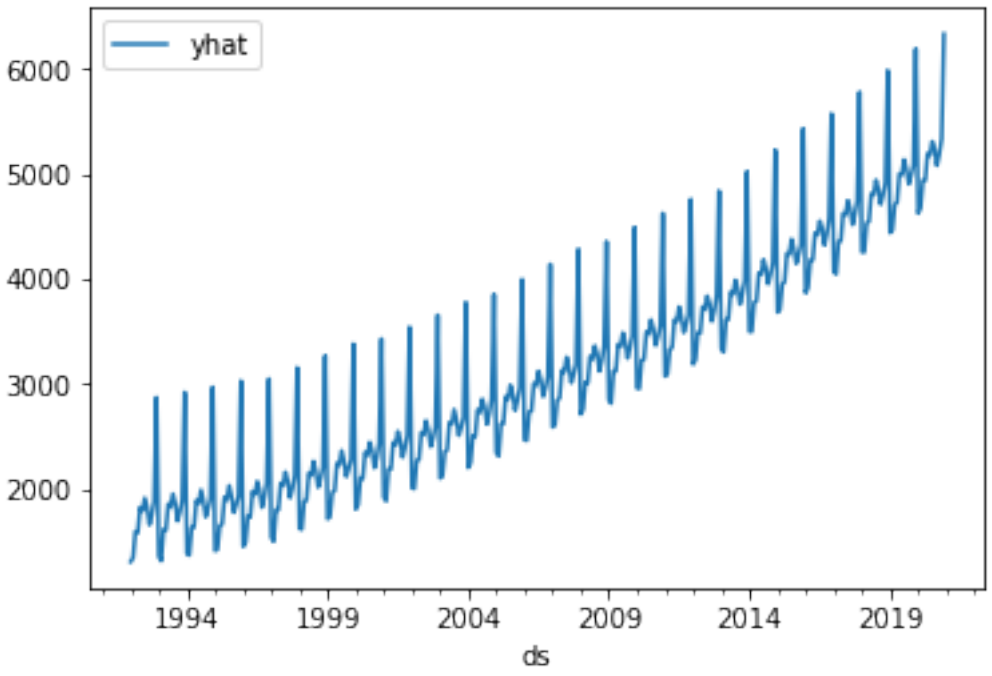
[18]:
```
import matplotlib.pyplot as plt
m.plot(forecast)
plt.xlim('2014-01-01','2022-01-01')
```

[18]: (735234.0, 738156.0)

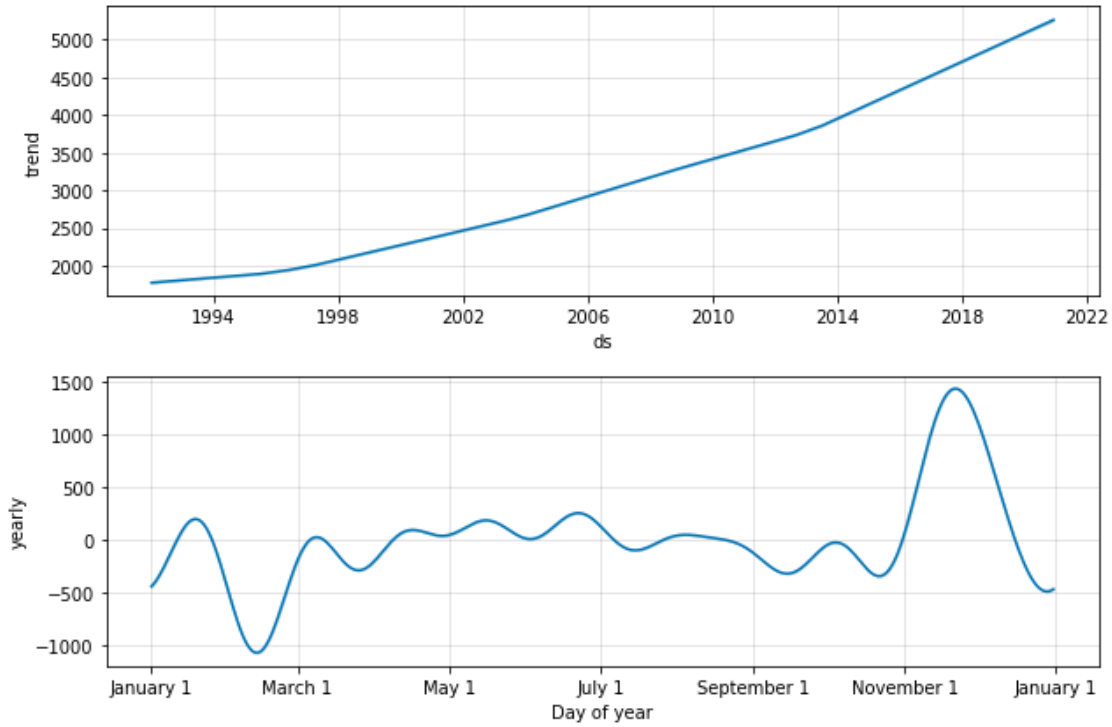


[19]:
```
forecast.plot(x='ds',y='yhat')
```

[19]: <matplotlib.axes._subplots.AxesSubplot at 0x1a23c48c88>



[20]:
```
m.plot_components(forecast);
```



0.11 Great Job!

[ ]: