In [1]:
```
from matplotlib import pyplot as plt
from matplotlib.dates import MonthLocator, num2date
from matplotlib.ticker import FuncFormatter
from prophet import Prophet
from prophet.diagnostics import cross_validation, performance_metrics
from prophet.plot import add_changepoints_to_plot

import pandas as pd
import numpy as np
import datetime as dt
from collections import defaultdict
import time
import datetime as dt
from pytz import timezone
tz = timezone('EST')
from tqdm import tqdm

from sklearn.metrics import mean_absolute_error, mean_absolute_percentage_error,

import seaborn as sns
%config InlineBackend.figure_format = 'retina'
%matplotlib inline
from matplotlib import pyplot as plt
from matplotlib import style
sns.set()
```

In [2]:
```
nh2011 = pd.ExcelFile(r"C:\Users\Rohan\Desktop\Big Data\Load Data\smd_hourly_2011
nh2011 = pd.read_excel(nh2011, 'NH')
nh2012 = pd.ExcelFile(r"C:\Users\Rohan\Desktop\Big Data\Load Data\smd_hourly_2011
nh2012 = pd.read_excel(nh2012, 'NH')
nh2013 = pd.ExcelFile(r"C:\Users\Rohan\Desktop\Big Data\Load Data\smd_hourly_2011
nh2013 = pd.read_excel(nh2013, 'NH')
nh2014 = pd.ExcelFile(r"C:\Users\Rohan\Desktop\Big Data\Load Data\smd_hourly_2011
nh2014 = pd.read_excel(nh2014, 'NH')
nh2015 = pd.ExcelFile(r"C:\Users\Rohan\Desktop\Big Data\Load Data\smd_hourly_2011
nh2015 = pd.read_excel(nh2015, 'NH')
nh2016 = pd.ExcelFile(r"C:\Users\Rohan\Desktop\Big Data\Load Data\smd_hourly_2011
nh2016 = pd.read_excel(nh2016, 'NH')
```

In [3]:
```
nh2017 = pd.read_excel(r"C:\Users\Rohan\Desktop\Big Data\Load Data\smd_hourly_201
nh2018 = pd.read_excel(r"C:\Users\Rohan\Desktop\Big Data\Load Data\smd_hourly_201
nh2019 = pd.read_excel(r"C:\Users\Rohan\Desktop\Big Data\Load Data\smd_hourly_201
nh2020 = pd.read_excel(r"C:\Users\Rohan\Desktop\Big Data\Load Data\smd_hourly_201
nh2021 = pd.read_excel(r"C:\Users\Rohan\Desktop\Big Data\Load Data\smd_hourly_201
nh2022 = pd.read_excel(r"C:\Users\Rohan\Desktop\Big Data\Load Data\smd_hourly_201
```

In [4]:
```
nh2022
```

Out[4]:

	Date	Hr_End	DA_Demand	RT_Demand	DA_LMP	DA_EC	DA_CC	DA_MLC	RT_LMP	RT_
0	2022-01-01	1	1005.6	1042.181	32.93	32.35	0.10	0.48	26.38	25.
1	2022-01-01	2	985.4	994.703	32.91	32.31	0.13	0.47	26.28	25.
2	2022-01-01	3	930.9	964.326	31.31	30.85	0.00	0.46	27.94	27.
3	2022-01-01	4	912.0	951.081	30.11	29.69	0.00	0.42	25.53	25.
4	2022-01-01	5	917.7	953.029	31.33	30.86	0.00	0.47	29.76	29.
...	...	...	...	...	...	...	...	...	...	...
739	2022-01-31	20	1683.1	1727.322	227.22	226.40	0.00	0.82	298.61	295.
740	2022-01-31	21	1585.7	1655.507	203.58	202.82	0.00	0.76	266.56	264.
741	2022-01-31	22	1456.6	1559.954	183.16	183.42	0.00	-0.26	253.85	251.
742	2022-01-31	23	1362.5	1452.523	180.39	179.93	0.00	0.46	191.90	190.
743	2022-01-31	24	1237.7	1370.635	190.67	190.54	0.00	0.13	191.20	189.

744 rows × 14 columns

In [6]:
```
val2011 = nh2011['DEMAND']
val2012 = nh2012['DEMAND']
val2013 = nh2013['DEMAND']
val2014 = nh2014['DEMAND']
val2015 = nh2015['DEMAND']
val2016 = nh2016['RT_Demand']
val2017 = nh2017['RT_Demand']
val2018 = nh2018['RT_Demand']
val2019 = nh2019['RT_Demand']
val2020 = nh2020['RT_Demand']
val2021 = nh2021['RT_Demand']
val2022 = nh2022['RT_Demand']
```

In [7]:
```
values = [val2011, val2012, val2013, val2014, val2015, val2016, val2017, val2018,
values_df = pd.concat(values, axis=0, ignore_index=False)
values_df = values_df.reset_index()
period = len(values_df)
```

In [8]:
```
rng = pd.date_range('2011-01-01', periods=period, freq='1H')
date_df = pd.DataFrame({ 'ds': rng})
date_df = date_df.reset_index()
```

In [9]:
```
frames = [date_df, values_df]
nh_load = pd.concat(frames, axis=1, ignore_index=False)
nh_load = nh_load.rename(columns={nh_load.columns[1]: 'ds', nh_load.columns[3]: '
frames2 = [nh_load['ds'], nh_load['y']]
nh_load = pd.concat(frames2, axis=1, ignore_index=False)
nh_load
```

Out[9]:

	ds	y
0	2011-01-01 00:00:00	1044.000
1	2011-01-01 01:00:00	985.000
2	2011-01-01 02:00:00	937.000
3	2011-01-01 03:00:00	914.000
4	2011-01-01 04:00:00	911.000
...	...	...
97171	2022-01-31 19:00:00	1727.322
97172	2022-01-31 20:00:00	1655.507
97173	2022-01-31 21:00:00	1559.954
97174	2022-01-31 22:00:00	1452.523
97175	2022-01-31 23:00:00	1370.635

97176 rows × 2 columns

In [10]:
```
model = Prophet(
        changepoint_prior_scale=0.5,
        seasonality_mode='multiplicative',
        interval_width=0.95,
    )
model.add_country_holidays(country_name='US')
```

Out[10]:
```
<prophet.forecaster.Prophet at 0x25a71276fa0>
```

In [11]:
```
model.fit(nh_load)
```

Out[11]:
```
<prophet.forecaster.Prophet at 0x25a71276fa0>
```

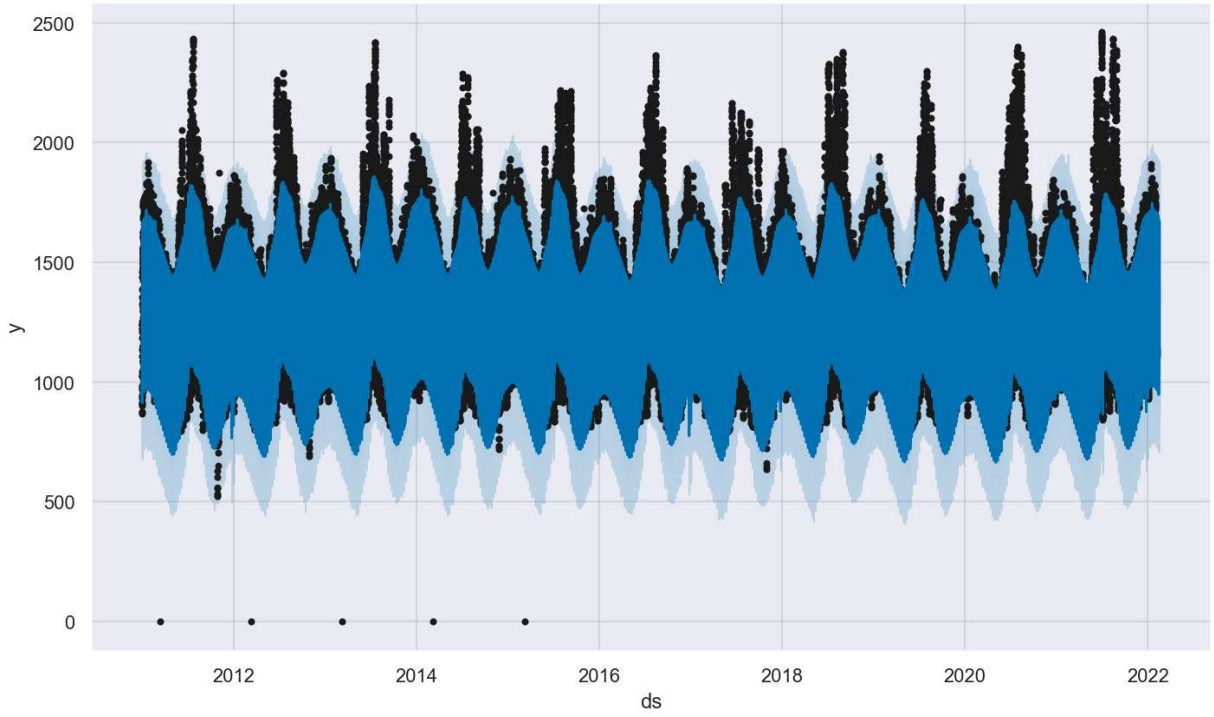
In [12]:
```
future_pd = model.make_future_dataframe(
        periods=365,
        freq='1H',
        include_history=True
    )

# make predictions
forecast_pd = model.predict(future_pd)
```

In [13]:
```
forecast_pd[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()
```

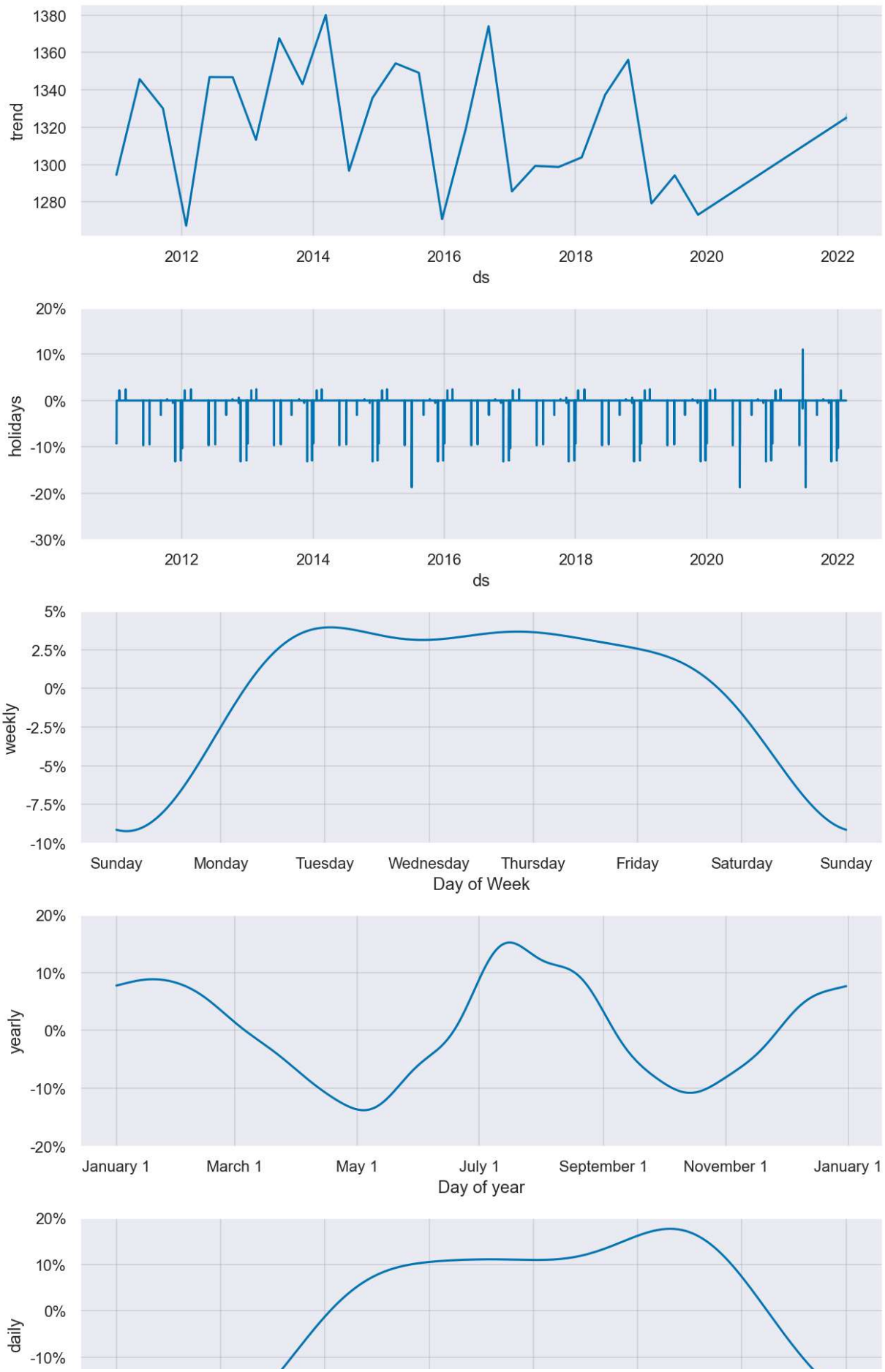
Out[13]:

	ds	yhat	yhat_lower	yhat_upper
97536	2022-02-16 00:00:00	1182.456664	936.096702	1420.472398
97537	2022-02-16 01:00:00	1130.074481	897.272045	1382.218908
97538	2022-02-16 02:00:00	1101.156882	859.554086	1335.352320
97539	2022-02-16 03:00:00	1101.677719	868.771191	1346.728065
97540	2022-02-16 04:00:00	1141.990520	910.164419	1388.278845

In [14]:
```
fig1 = model.plot(forecast_pd)
```



In [15]:
```
fig2 = model.plot_components(forecast_pd)
```



In [16]:
```
forecast_pd
```

Out[16]:

	ds	trend	yhat_lower	yhat_upper	trend_lower	trend_upper	Christmas Day	Chris Day_
0	2011-01-01 00:00:00	1294.464824	764.192191	1240.217515	1294.464824	1294.464824	0.0	
1	2011-01-01 01:00:00	1294.481304	712.119613	1181.324228	1294.481304	1294.481304	0.0	
2	2011-01-01 02:00:00	1294.497785	683.595736	1176.224546	1294.497785	1294.497785	0.0	
3	2011-01-01 03:00:00	1294.514265	677.266281	1158.822168	1294.514265	1294.514265	0.0	
4	2011-01-01 04:00:00	1294.530746	713.115710	1210.343837	1294.530746	1294.530746	0.0	
...	...	...	...	...	...	...	...	
97536	2022-02-16 00:00:00	1324.972282	936.096702	1420.472398	1323.260162	1327.330826	0.0	
97537	2022-02-16 01:00:00	1324.974901	897.272045	1382.218908	1323.256722	1327.355235	0.0	
97538	2022-02-16 02:00:00	1324.977519	859.554086	1335.352320	1323.253283	1327.371442	0.0	
97539	2022-02-16 03:00:00	1324.980138	868.771191	1346.728065	1323.249843	1327.380809	0.0	
97540	2022-02-16 04:00:00	1324.982756	910.164419	1388.278845	1323.243724	1327.390050	0.0	

97541 rows × 73 columns

In [ ]:
```

```