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]:
```
me2011 = pd.ExcelFile(r"C:\Users\Rohan\Desktop\Big Data\Load Data\smd_hourly_2011
me2011 = pd.read_excel(me2011, 'ME')
me2012 = pd.ExcelFile(r"C:\Users\Rohan\Desktop\Big Data\Load Data\smd_hourly_2011
me2012 = pd.read_excel(me2012, 'ME')
me2013 = pd.ExcelFile(r"C:\Users\Rohan\Desktop\Big Data\Load Data\smd_hourly_2011
me2013 = pd.read_excel(me2013, 'ME')
me2014 = pd.ExcelFile(r"C:\Users\Rohan\Desktop\Big Data\Load Data\smd_hourly_2011
me2014 = pd.read_excel(me2014, 'ME')
me2015 = pd.ExcelFile(r"C:\Users\Rohan\Desktop\Big Data\Load Data\smd_hourly_2011
me2015 = pd.read_excel(me2015, 'ME')
me2016 = pd.ExcelFile(r"C:\Users\Rohan\Desktop\Big Data\Load Data\smd_hourly_2011
me2016 = pd.read_excel(me2016, 'ME')
```

In [3]:
```
me2017 = pd.read_excel(r"C:\Users\Rohan\Desktop\Big Data\Load Data\smd_hourly_201
me2018 = pd.read_excel(r"C:\Users\Rohan\Desktop\Big Data\Load Data\smd_hourly_201
me2019 = pd.read_excel(r"C:\Users\Rohan\Desktop\Big Data\Load Data\smd_hourly_201
me2020 = pd.read_excel(r"C:\Users\Rohan\Desktop\Big Data\Load Data\smd_hourly_201
me2021 = pd.read_excel(r"C:\Users\Rohan\Desktop\Big Data\Load Data\smd_hourly_201
me2022 = pd.read_excel(r"C:\Users\Rohan\Desktop\Big Data\Load Data\smd_hourly_201
```

In [4]:
```
me2022
```

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	867.0	1152.722	32.86	32.35	0.23	0.28	26.59	25.
1	2022-01-01	2	978.2	1116.946	33.00	32.31	0.28	0.41	26.50	25.
2	2022-01-01	3	923.3	1093.610	31.52	30.85	0.00	0.67	28.11	27.
3	2022-01-01	4	906.2	1085.773	30.30	29.69	0.00	0.61	25.67	25.
4	2022-01-01	5	921.8	1099.982	31.53	30.86	0.00	0.67	29.99	29.
...	...	...	...	...	...	...	...	...	...	
739	2022-01-31	20	1633.2	1846.799	222.87	226.40	0.00	-3.53	295.56	295.
740	2022-01-31	21	1522.4	1777.463	199.97	202.82	0.00	-2.85	262.68	264.
741	2022-01-31	22	1325.4	1699.196	180.35	183.42	0.00	-3.07	250.05	251.
742	2022-01-31	23	1177.7	1555.524	177.46	179.93	0.00	-2.47	188.78	190.
743	2022-01-31	24	1087.4	1283.547	188.77	190.54	0.00	-1.77	188.83	189.

744 rows × 14 columns

In [5]:
```
val2011 = me2011['DEMAND']
val2012 = me2012['DEMAND']
val2013 = me2013['DEMAND']
val2014 = me2014['DEMAND']
val2015 = me2015['DEMAND']
val2016 = me2016['RT_Demand']
val2017 = me2017['RT_Demand']
val2018 = me2018['RT_Demand']
val2019 = me2019['RT_Demand']
val2020 = me2020['RT_Demand']
val2021 = me2021['RT_Demand']
val2022 = me2022['RT_Demand']
```

In [6]:
```
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 [7]:
```
rng = pd.date_range('2011-01-01', periods=period, freq='1H')
date_df = pd.DataFrame({ 'ds': rng})
date_df = date_df.reset_index()
```

In [8]:
```
frames = [date_df, values_df]
me_load = pd.concat(frames, axis=1, ignore_index=False)
me_load = me_load.rename(columns={me_load.columns[1]: 'ds', me_load.columns[3]: '
frames2 = [me_load['ds'], me_load['y']]
me_load = pd.concat(frames2, axis=1, ignore_index=False)
me_load
```

Out[8]:

	ds	y
0	2011-01-01 00:00:00	1048.000
1	2011-01-01 01:00:00	1000.000
2	2011-01-01 02:00:00	964.000
3	2011-01-01 03:00:00	954.000
4	2011-01-01 04:00:00	960.000
...	...	...
97171	2022-01-31 19:00:00	1846.799
97172	2022-01-31 20:00:00	1777.463
97173	2022-01-31 21:00:00	1699.196
97174	2022-01-31 22:00:00	1555.524
97175	2022-01-31 23:00:00	1283.547

97176 rows × 2 columns

In [9]:
```
model = Prophet(
        changepoint_prior_scale=0.5,
        seasonality_mode='multiplicative',
        interval_width=0.95,
    )
model.add_country_holidays(country_name='US')
```

Out[9]:
```
<prophet.forecaster.Prophet at 0x20f6219be20>
```

In [10]:
```
model.fit(me_load)
```

Out[10]:
```
<prophet.forecaster.Prophet at 0x20f6219be20>
```

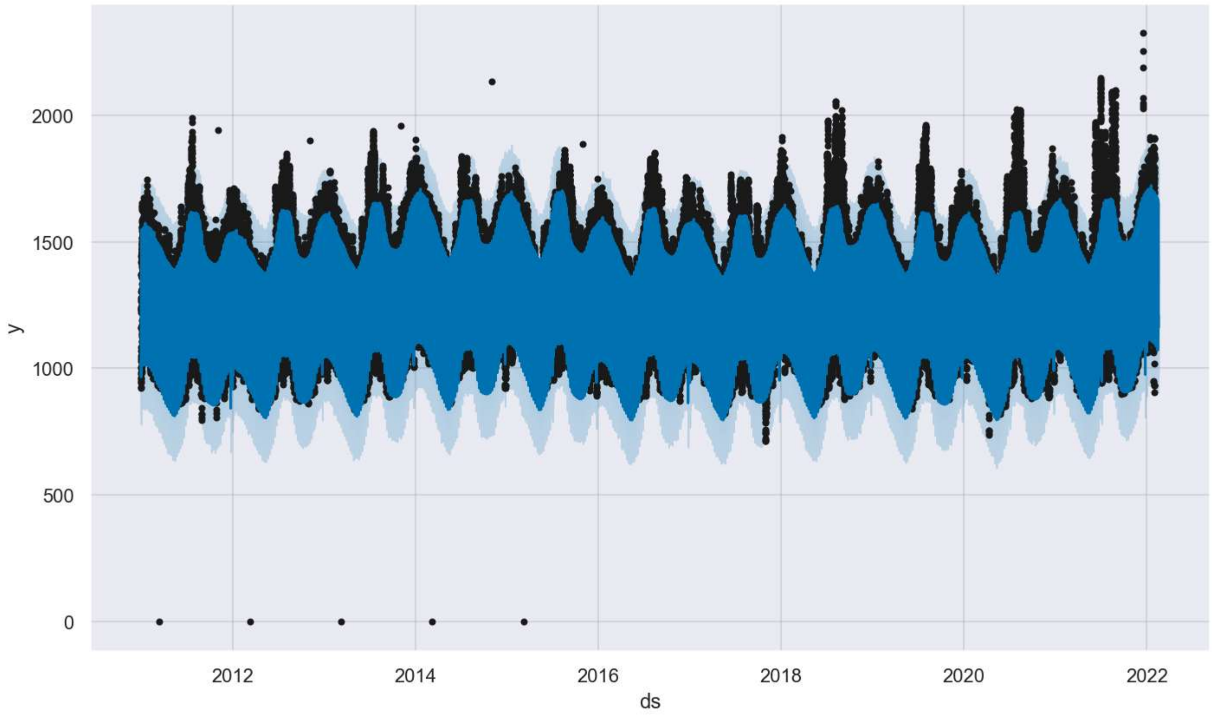
In [11]:
```
future_pd = model.make_future_dataframe(
        periods=365,
        freq='1H',
        include_history=True
    )

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

In [12]:
```
forecast_pd[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()
```

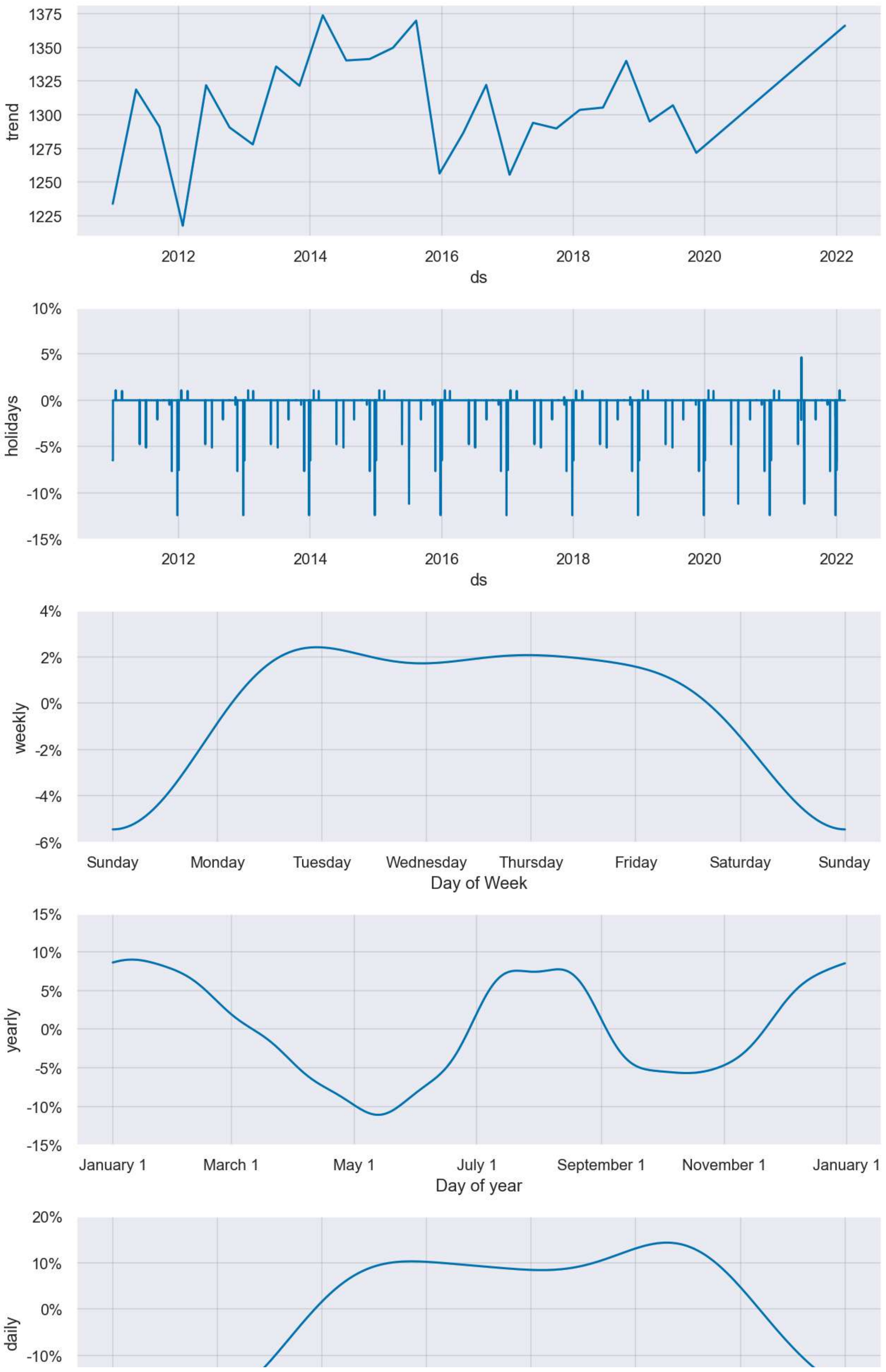
Out[12]:

	ds	yhat	yhat_lower	yhat_upper
97536	2022-02-16 00:00:00	1222.310143	1054.757560	1401.791215
97537	2022-02-16 01:00:00	1181.339685	1006.014964	1343.711103
97538	2022-02-16 02:00:00	1160.001157	997.737491	1337.266448
97539	2022-02-16 03:00:00	1165.047049	983.699111	1348.152193
97540	2022-02-16 04:00:00	1207.842258	1041.070343	1384.551975

In [13]:
```
fig1 = model.plot(forecast_pd)
```



In [14]:
```
fig2 = model.plot_components(forecast_pd)
```



-20%
-30%
00:00:00
03:25:42
06:51:25
10:17:08
13:42:51
17:08:34
20:34:17
00:00:00
Hour of day

In [15]:
```
forecast_pd
```

Out[15]:

	ds	trend	yhat_lower	yhat_upper	trend_lower	trend_upper	Christmas Day	Chr Day
0	2011-01-01 00:00:00	1233.930190	843.307469	1191.216056	1233.930190	1233.930190	0.0	
1	2011-01-01 01:00:00	1233.957439	811.022262	1155.098201	1233.957439	1233.957439	0.0	
2	2011-01-01 02:00:00	1233.984687	797.352995	1149.801963	1233.984687	1233.984687	0.0	
3	2011-01-01 03:00:00	1234.011935	778.982171	1141.598839	1234.011935	1234.011935	0.0	
4	2011-01-01 04:00:00	1234.039184	831.992126	1186.027033	1234.039184	1234.039184	0.0	
...	...	...	...	...	...	...	...	
97536	2022-02-16 00:00:00	1366.016111	1054.757560	1401.791215	1365.572354	1366.656980	0.0	
97537	2022-02-16 01:00:00	1366.020876	1006.014964	1343.711103	1365.575664	1366.663952	0.0	
97538	2022-02-16 02:00:00	1366.025640	997.737491	1337.266448	1365.565120	1366.670923	0.0	
97539	2022-02-16 03:00:00	1366.030404	983.699111	1348.152193	1365.542609	1366.682428	0.0	
97540	2022-02-16 04:00:00	1366.035168	1041.070343	1384.551975	1365.537268	1366.701257	0.0	

97541 rows × 73 columns

In [ ]: