```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns


# import zipfile
# file = zipfile.ZipFile('/content/household_power_consumption.zip', 'r')
# file.extractall('data')
power_data = pd.read_csv('/content/data/household_power_consumption.txt', sep = ';', header=0
                 low_memory=False, infer_datetime_format=True,
                 parse_dates={'datetime':[0,1]}, index_col=['datetime'])


power_data.head()
```

datetime	Global_active_power	Global_reactive_power	Voltage	Global_intensity	Sub_me
2006-12-16 17:24:00	4.216	0.418	234.840	18.400	
2006-12-16 17:25:00	5.360	0.436	233.630	23.000	
2006-12-16 17:26:00	5.374	0.498	233.290	23.000	
2006-12-16 17:27:00	5.388	0.502	233.740	23.000	
2006-12-16 17:28:00	3.666	0.528	235.680	15.800	

```
power_data.replace('?', 'nan', inplace=True)
power_data = power_data.astype('float32')


power_data.isna().sum()
```

```
Global_active_power      25979
Global_reactive_power    25979
Voltage                  25979
Global_intensity         25979
```

```
Sub_metering_1          25979
Sub_metering_2          25979
Sub_metering_3          25979
dtype: int64
```

```
power_data.describe()
```

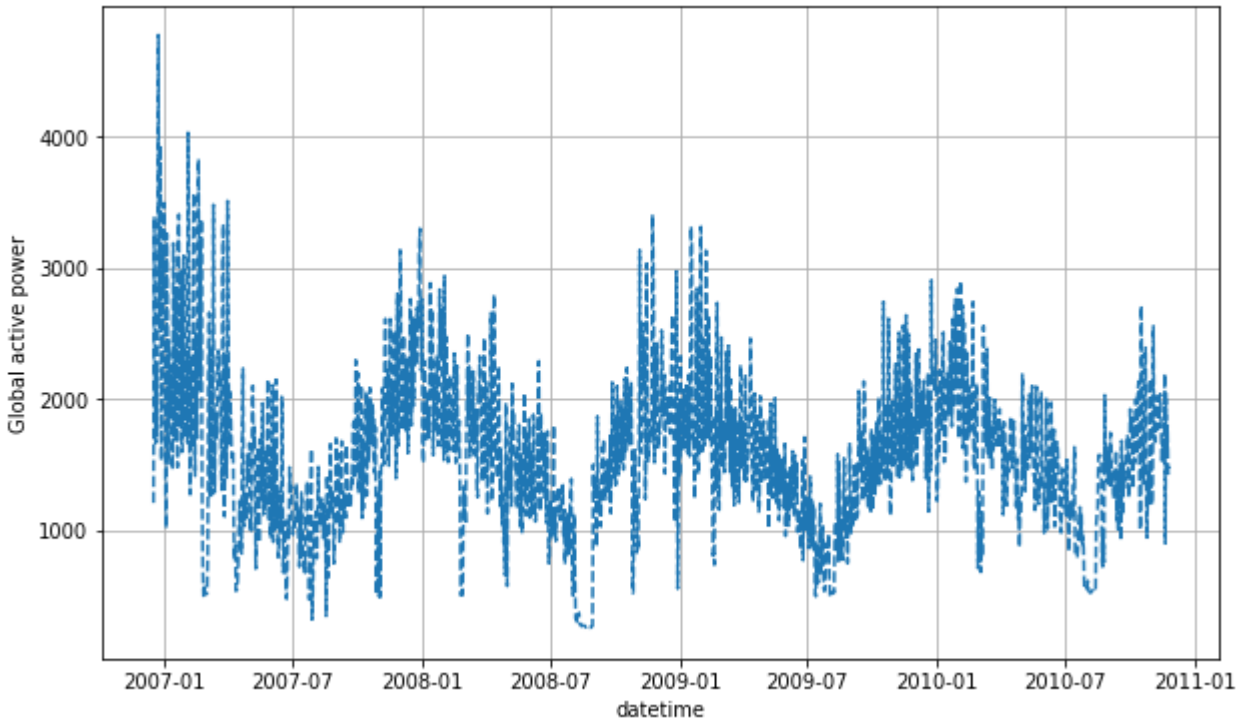
	Global_active_power	Global_reactive_power	Voltage	Global_intensity	Sub_
count	2.049280e+06	2.049280e+06	2.049280e+06	2.049280e+06	2.
mean	1.091631e+00	1.237042e-01	2.433813e+02	4.629239e+00	1.
std	1.057005e+00	1.128308e-01	4.114049e+00	4.440444e+00	6.
min	7.600000e-02	0.000000e+00	2.232000e+02	2.000000e-01	0.
25%	3.080000e-01	4.800000e-02	2.389900e+02	1.400000e+00	0.
50%	6.020000e-01	1.000000e-01	2.410100e+02	2.600000e+00	0.
75%	1.528000e+00	1.940000e-01	2.428900e+02	6.400000e+00	0.
max	1.112200e+01	1.390000e+00	2.541500e+02	4.840000e+01	8.

```
power_data = power_data.fillna(power_data.mean())

power_data_daily = power_data.resample('D').sum()
power_data_daily.head()
```

datetime	Global_active_power	Global_reactive_power	Voltage	Global_intensity	S
2006-12-16	1209.176025	34.922001	93552.53125	5180.799805	
2006-12-17	3390.459961	226.005997	345725.31250	14398.599609	
2006-12-18	2203.825928	161.792007	347373.62500	9247.200195	
2006-12-19	1666.193970	150.942001	348479.00000	7094.000000	
2006-12-20	2225.748047	160.998001	348923.62500	9313.000000	

```
plt.figure(figsize=(10,6))
plt.plot(power_data_daily.index, power_data_daily.Global_active_power, '--')
plt.grid()
```

```
plt.xlabel('datetime')
plt.ylabel('Global active power')
plt.show()
```




```
import scipy
import scipy.stats
from scipy.stats import pearsonr

corr2,_ = pearsonr(power_data_daily.Voltage,power_data_daily.Global_intensity)
corr2
```

```
0.05221350712869458
```

```
train_data = power_data_daily.iloc[:1077,:]
test_data = power_data_daily.iloc[1077:,:]

train_data.reset_index(inplace=True)
train_data.head()
```

	datetime	Global_active_power	Global_reactive_power	Voltage	Global_intensity
0	2006-12-16	1209.176025	34.922001	93552.53125	5180.799805
	2006-12				

```
power_data_train = train_data.iloc[:,:2]
power_data_train = power_data_train.rename(columns={"datetime": "ds", "Global_active_power":

power_data_train.head()
```

	ds	y
0	2006-12-16	1209.176025
1	2006-12-17	3390.459961
2	2006-12-18	2203.825928
3	2006-12-19	1666.193970
4	2006-12-20	2225.748047

```
test_data.reset_index(inplace = True)
test_data.head()
```

	datetime	Global_active_power	Global_reactive_power	Voltage	Global_intensity
0	2009-11-27	1380.026001	133.052002	348276.68750	5704.000000
1	2009-11-28	1858.949951	225.218002	347898.87500	7784.799805
2	2009-11-29	1650.962036	191.056000	350364.56250	6878.399902
3	2009-11-30	1745.189941	153.382004	351020.59375	7232.200195
4	2009-12-01	1756.378052	141.873993	349391.75000	7312.799805

```
power_data_test = test_data.iloc[:,:2]
power_data_test = power_data_test.rename(columns={"datetime": "ds", "Global_active_power": "y

power_data_test.head()
```

	ds	y
0	2009-11-27	1380.026001
1	2009-11-28	1858.949951
2	2009-11-29	1650.962036

```
from fbprophet import Prophet

model = Prophet(daily_seasonality=True)
model.fit(power_data_train)

forecast = model.predict(power_data_test)
forecast.head()
```

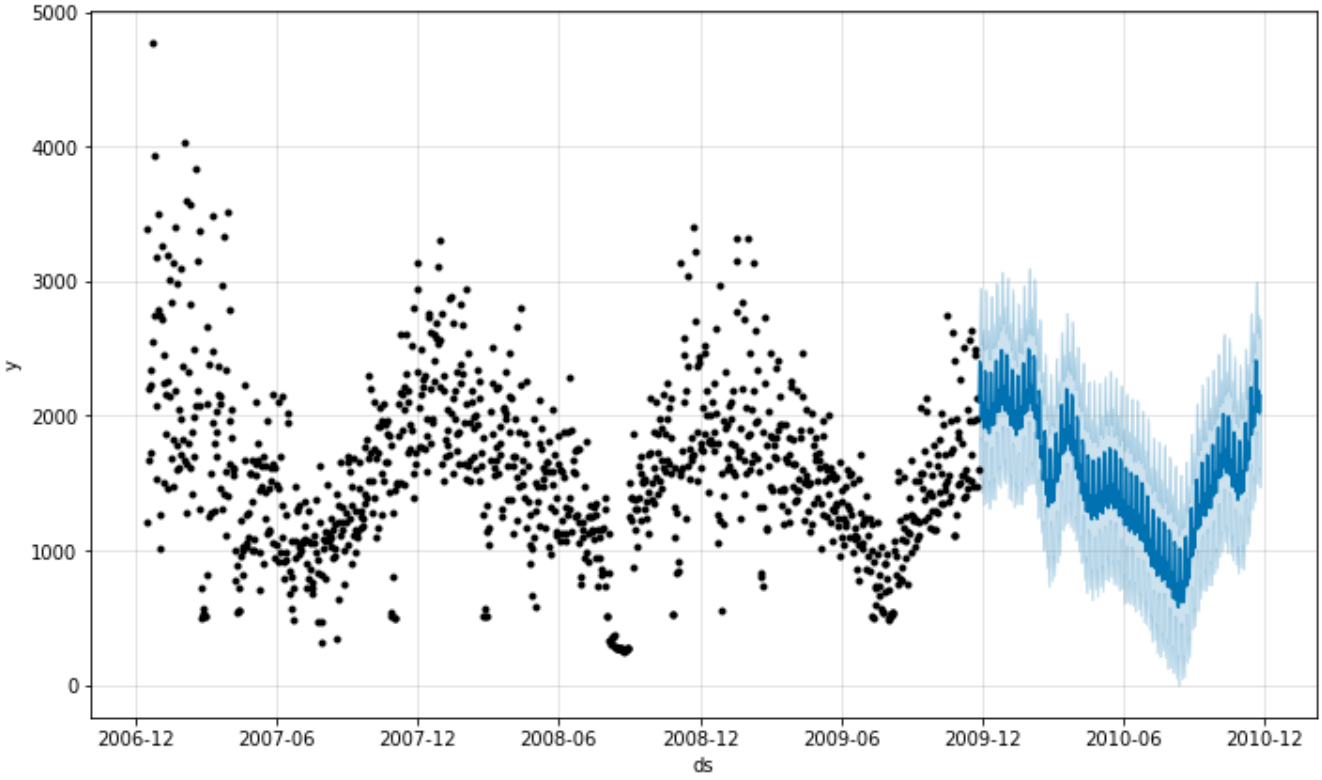
```
INFO:numexpr.utils:NumExpr defaulting to 2 threads.
```

	ds	trend	yhat_lower	yhat_upper	trend_lower	trend_upper	additive_terms
0	2009-11-27	1227.555640	1525.508146	2645.730352	1227.555640	1227.555640	870.93253(
1	2009-11-28	1227.677685	1838.838351	2951.679057	1227.677685	1227.677685	1174.16411
2	2009-11-29	1227.799730	1801.542553	2942.782066	1227.799730	1227.799730	1148.11146(
3	2009-11-30	1227.921775	1430.394478	2530.522450	1227.921775	1227.921775	748.013775
4	2009-12-01	1228.043820	1472.317478	2612.664234	1228.043820	1228.043820	868.831712

```
forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper', 'trend', 'trend_lower', 'trend_upper']].h
```

	ds	yhat	yhat_lower	yhat_upper	trend	trend_lower	trend_upper
0	2009-11-27	2098.488170	1525.508146	2645.730352	1227.555640	1227.555640	1227.555640
1	2009-11-28	2401.841796	1838.838351	2951.679057	1227.677685	1227.677685	1227.677685
2	2009-11-29	2375.911190	1801.542553	2942.782066	1227.799730	1227.799730	1227.799730
3	2009-	1975.935551	1430.394478	2530.522450	1227.921775	1227.921775	1227.921775

```
model.plot(forecast)
plt.show()
```



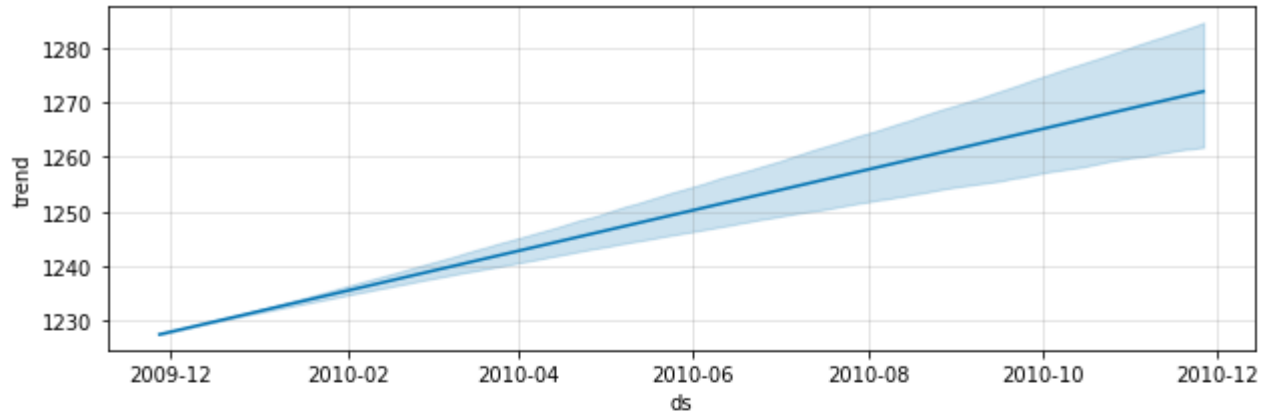

```
from sklearn import metrics

def Metric(y_true,y_pred):
  y_true,y_pred = np.array(y_true), np.array(y_pred)
  return np.mean(np.abs((y_true - y_pred)/y_true)) *100

MAPE = Metric(power_data_test['y'],forecast['yhat'])
round(MAPE,2)
```

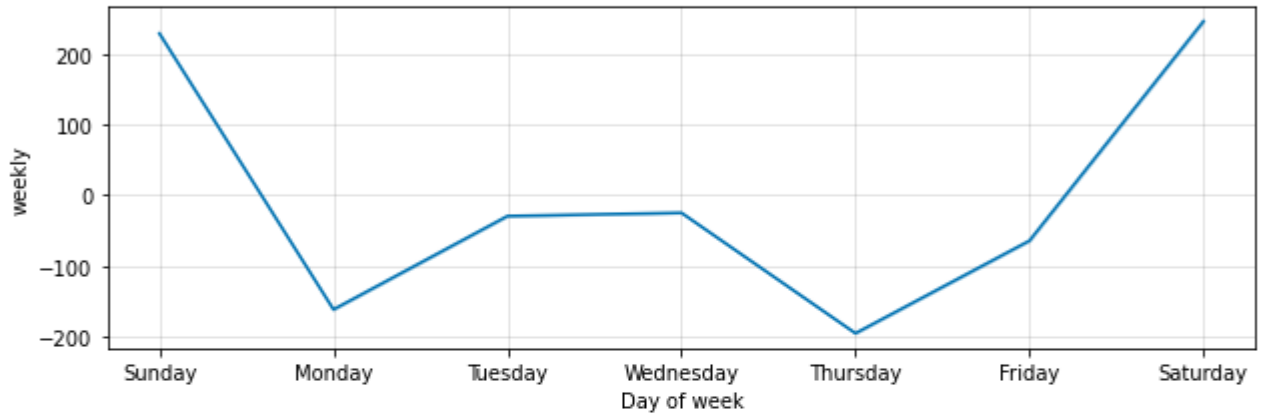
```
20.81
```

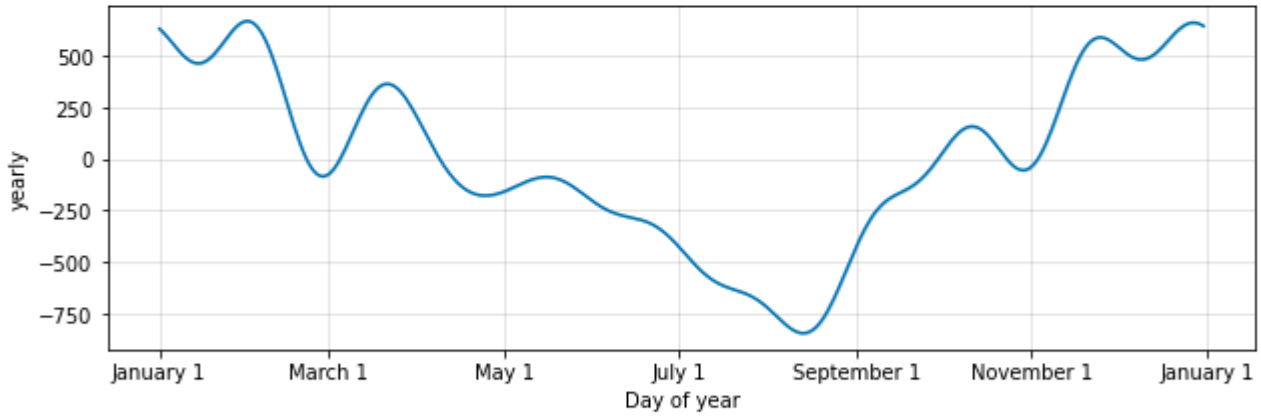
```
RMSE = np.sqrt(metrics.mean_squared_error(power_data_test['y'],forecast['yhat']))
round(RMSE,4)
```

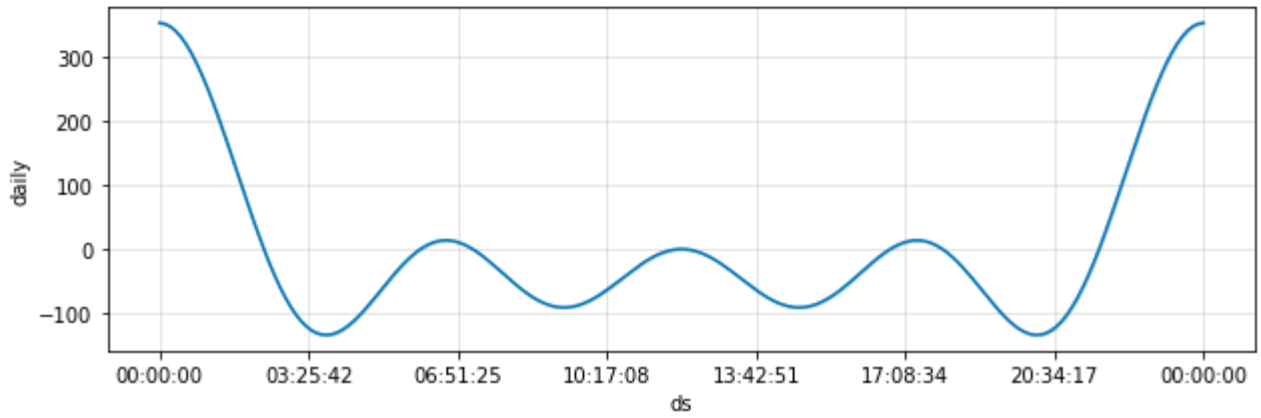
```
374.6036
```

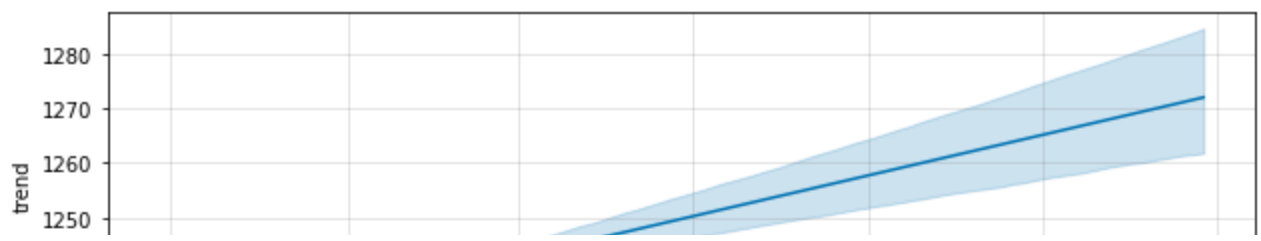
```
model.plot_components(forecast)
```


trend
1280
1270
1260
1250
1240
1230
2009-12
2010-02
2010-04
2010-06
2010-08
2010-10
2010-12
ds


weekly
200
100
0
−100
−200
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Day of week


yearly
500
250
0
−250
−500
−750
January 1
March 1
May 1
July 1
September 1
November 1
January 1
Day of year


daily
300
200
100
0
−100
00:00:00
03:25:42
06:51:25
10:17:08
13:42:51
17:08:34
20:34:17
00:00:00
ds


trend
1280
1270
1260
1250


```
power_data_daily.reset_index(inplace=True)
power_data_daily.head()
```

	datetime	Global_active_power	Global_reactive_power	Voltage	Global_intensity
0	2006-12-16	1209.176025	34.922001	93552.53125	5180.799805
1	2006-12-17	3390.459961	226.005997	345725.31250	14398.599609
2	2006-12-18	2203.825928	161.792007	347373.62500	9247.200195
3	2006-12-19	1666.193970	150.942001	348479.00000	7094.000000
4	2006-12-20	2225.748047	160.998001	348923.62500	9313.000000

```
new_power_daily = power_data_daily.rename(columns = {'datetime':'ds','Global_active_power':'y
                                                     'Global_reactive_power':'add1','Voltage':'add2',
```

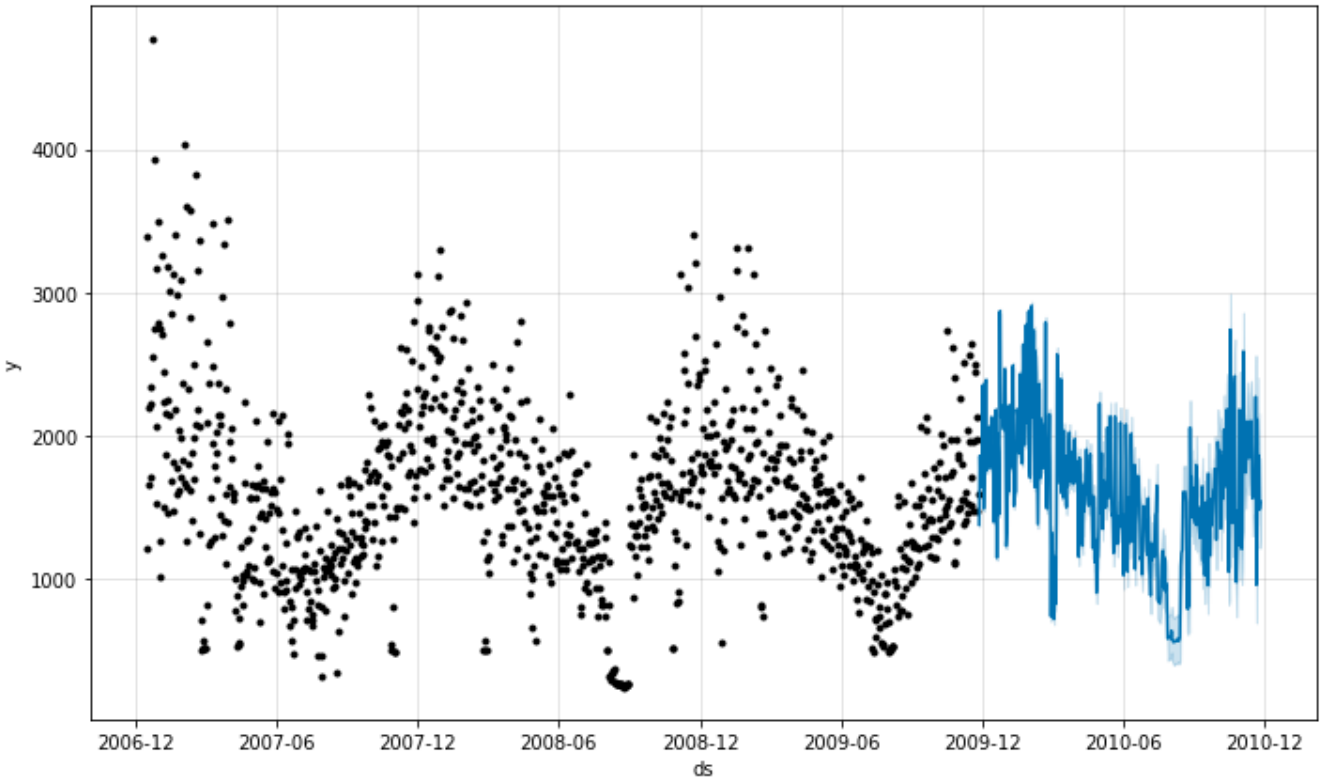
```
                                  'Global_intensity':'add3','Sub_metering_1':'add4',
                                 'Sub_metering_2':'add5','Sub_metering_3':'add6'})
new_power_daily.head()
```

	ds	y	add1	add2	add3	add4	add5	add6
0	2006-12-16	1209.176025	34.922001	93552.53125	5180.799805	0.0	546.0	4926.0
1	2006-12-17	3390.459961	226.005997	345725.31250	14398.599609	2033.0	4187.0	13341.0
2	2006-12-18	2203.825928	161.792007	347373.62500	9247.200195	1063.0	2621.0	14018.0
3	2006-12-	1666.193970	150.942001	348479.00000	7094.000000	839.0	7602.0	6197.0

```
new_train = new_power_daily.iloc[:1077,:]
new_test = new_power_daily.iloc[1077:,:]


model = Prophet(daily_seasonality=True)
model.add_regressor('add1')
model.add_regressor('add2')
model.add_regressor('add3')
model.add_regressor('add4')
model.add_regressor('add5')
model.add_regressor('add6')

model = model.fit(new_train)
new_forecast = model.predict(new_test)
new_forecast.head()
```

	ds	trend	yhat_lower	yhat_upper	trend_lower	trend_upper	add1	add
0	2009-11-27	1310.545634	1366.358923	1397.136463	1310.545634	1310.545634	7.518641	7
1	2009-11-28	1310.665825	1846.408260	1878.570534	1310.665825	1310.665825	-8.859759	-8
2	2009-11-29	1310.786015	1628.646876	1659.854577	1310.786015	1310.786015	-2.788985	-2
3	2009-11-30	1310.906205	1725.942049	1758.622995	1310.906205	1310.906205	3.905889	3
4	2009-12-01	1311.026396	1745.721688	1777.595947	1311.026396	1311.026396	5.950925	5

```
model.plot(new_forecast)
plt.show()
```



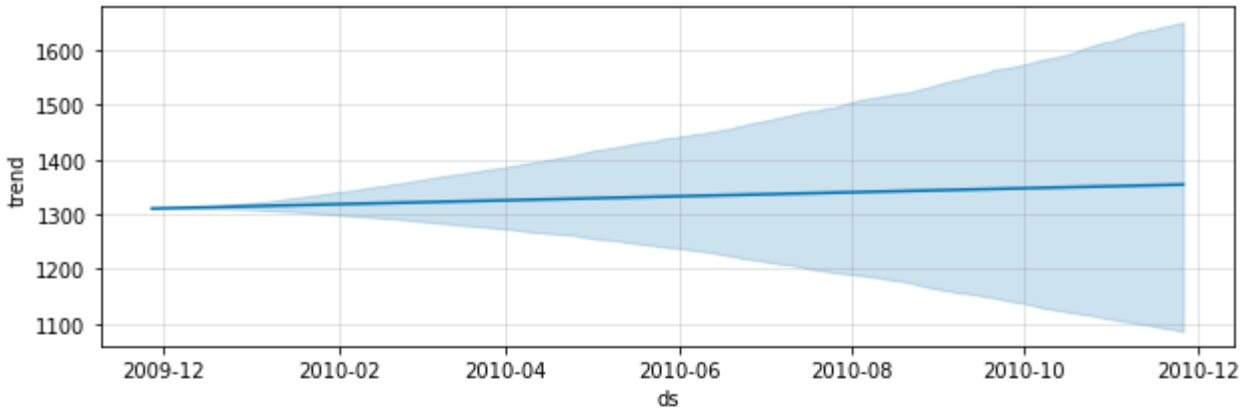

```
def Metric(y_true,y_pred):
  y_true,y_pred = np.array(y_true), np.array(y_pred)
  return np.mean(np.abs((y_true - y_pred)/y_true)) *100

MAPE = Metric(new_test['y'],new_forecast['yhat'])
round(MAPE,2)
```

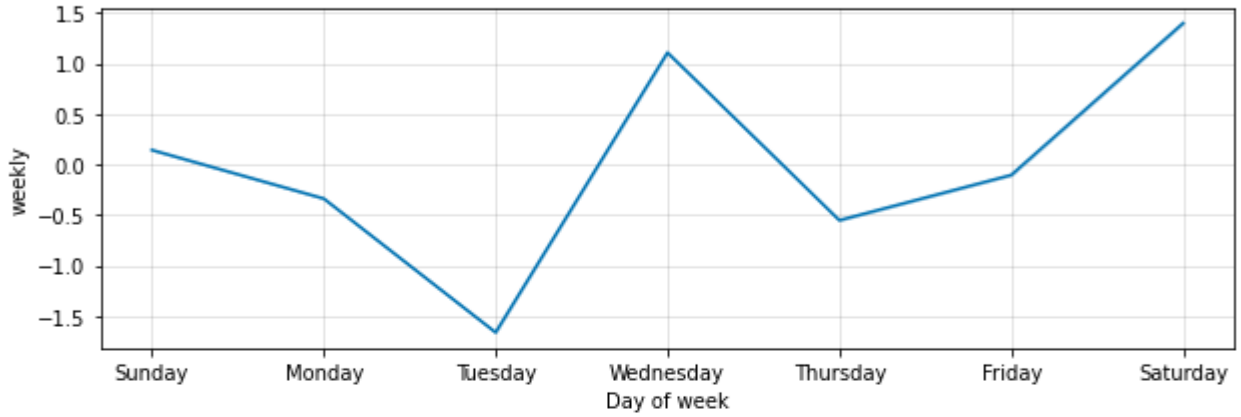
```
2.61
```

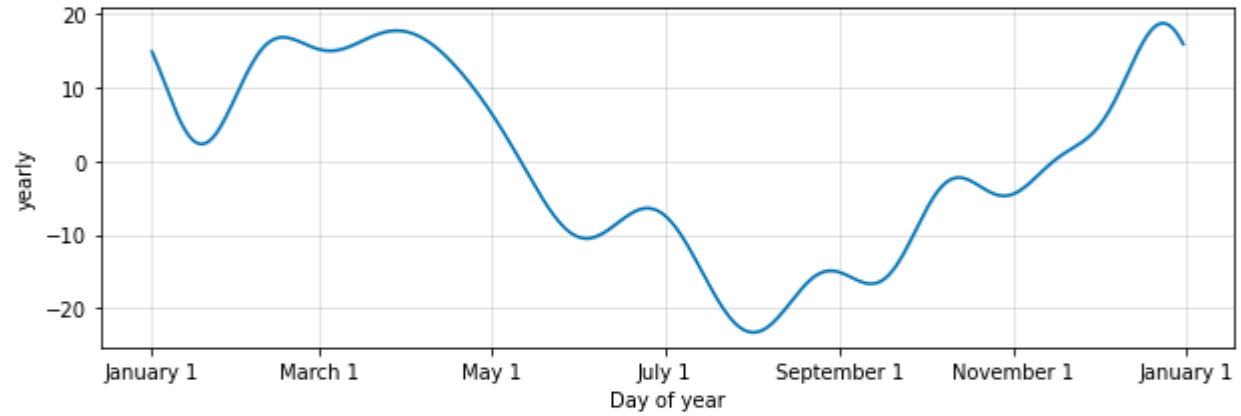
```
RMSE = np.sqrt(metrics.mean_squared_error(new_test['y'],new_forecast['yhat']))
round(RMSE,4)
```

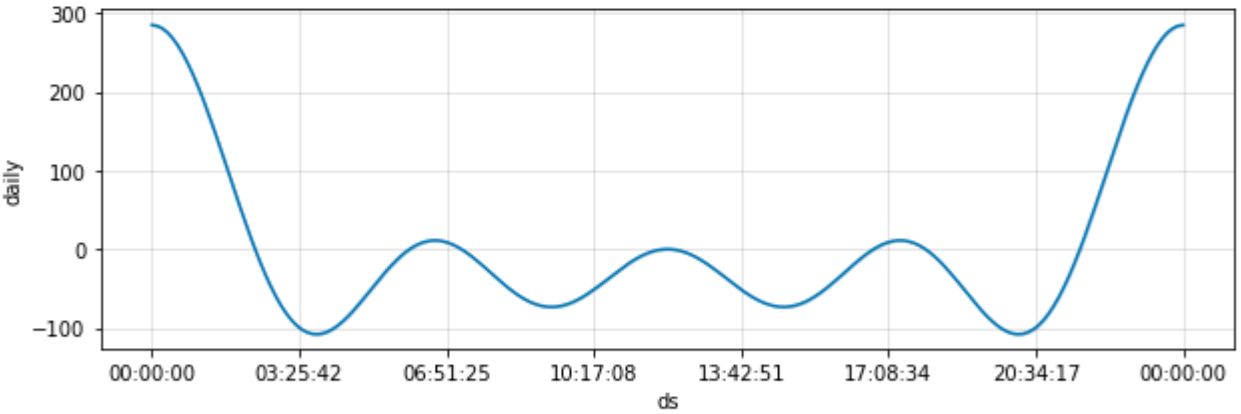
```
38.9406
```

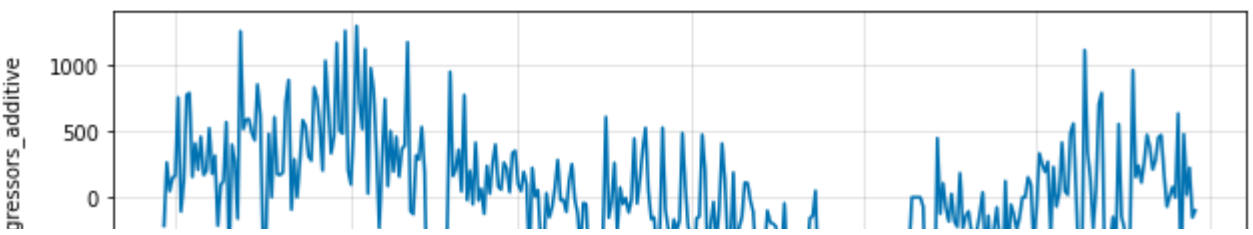
```
model.plot_components(new_forecast)
```


trend
1600
1500
1400
1300
1200
1100
2009-12
2010-02
2010-04
2010-06
2010-08
2010-10
2010-12
ds


weekly
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Day of week


yearly
20
10
0
−10
−20
January 1
March 1
May 1
July 1
September 1
November 1
January 1
Day of year


daily
300
200
100
0
−100
00:00:00
03:25:42
06:51:25
10:17:08
13:42:51
17:08:34
20:34:17
00:00:00
ds


gressors_additive
1000
500
0