
Forecasting
Energy
Demand

Who needs this service most?

Energy Suppliers

## Why do you need this service?

Energy as a resource constraint



Understand energy demand with data-driven

Provide optimal energy supply

# Choosing the right dataset

**French household between 2007-2010**

Power consumption

- Voltage
- Reactive power
- Power in the kitchen
- etc.

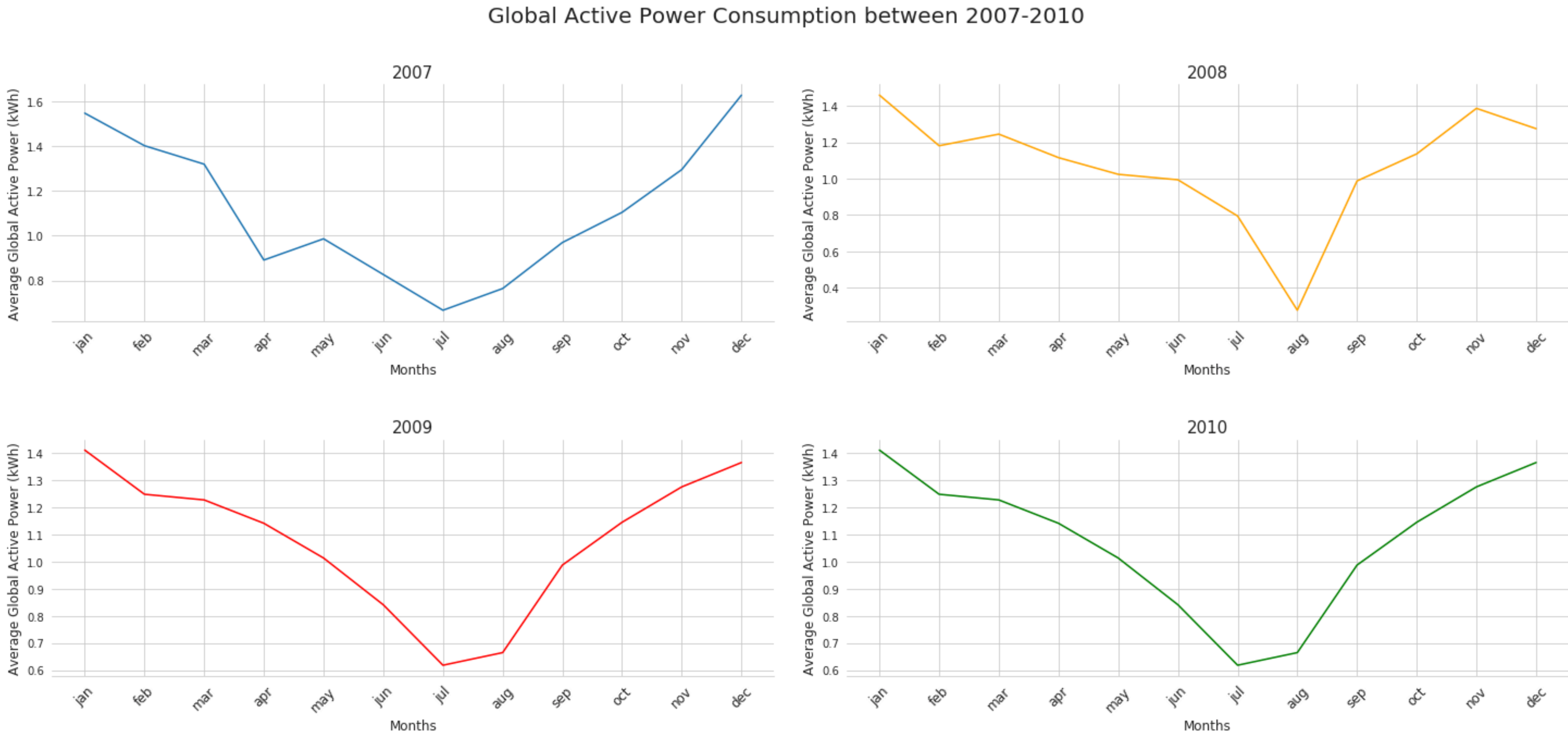
# Tackling the problem

# Energy demand has a clear seasonal trend across all four years

More energy is consumed during colder months

Less energy is consumed during the warmer months

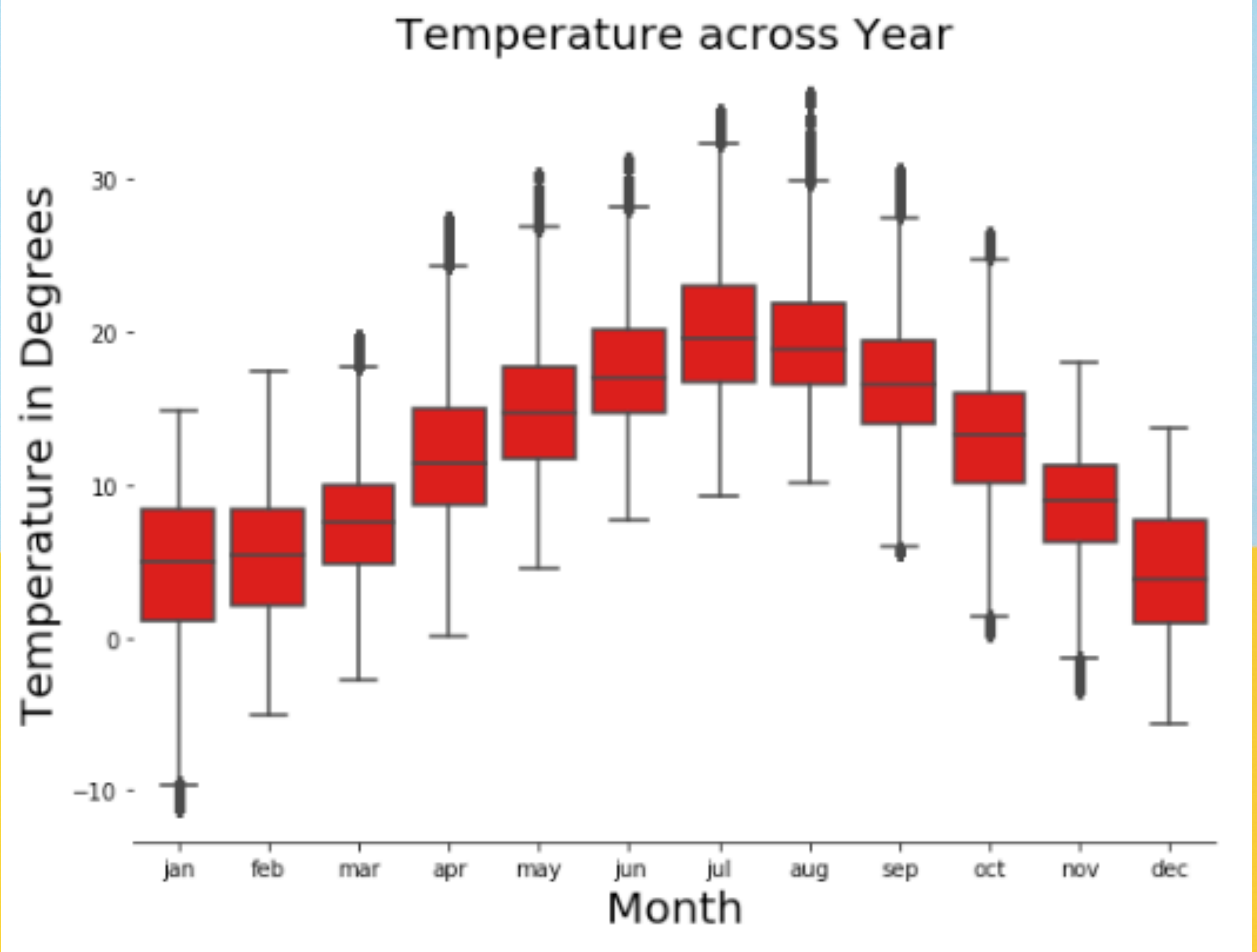
Temperature appears to have an effect on energy usage



## Temperatures across the year vary quite considerably

Average temperature in winter months hovers around 5°

Average temperature in summer months is nearly 20°

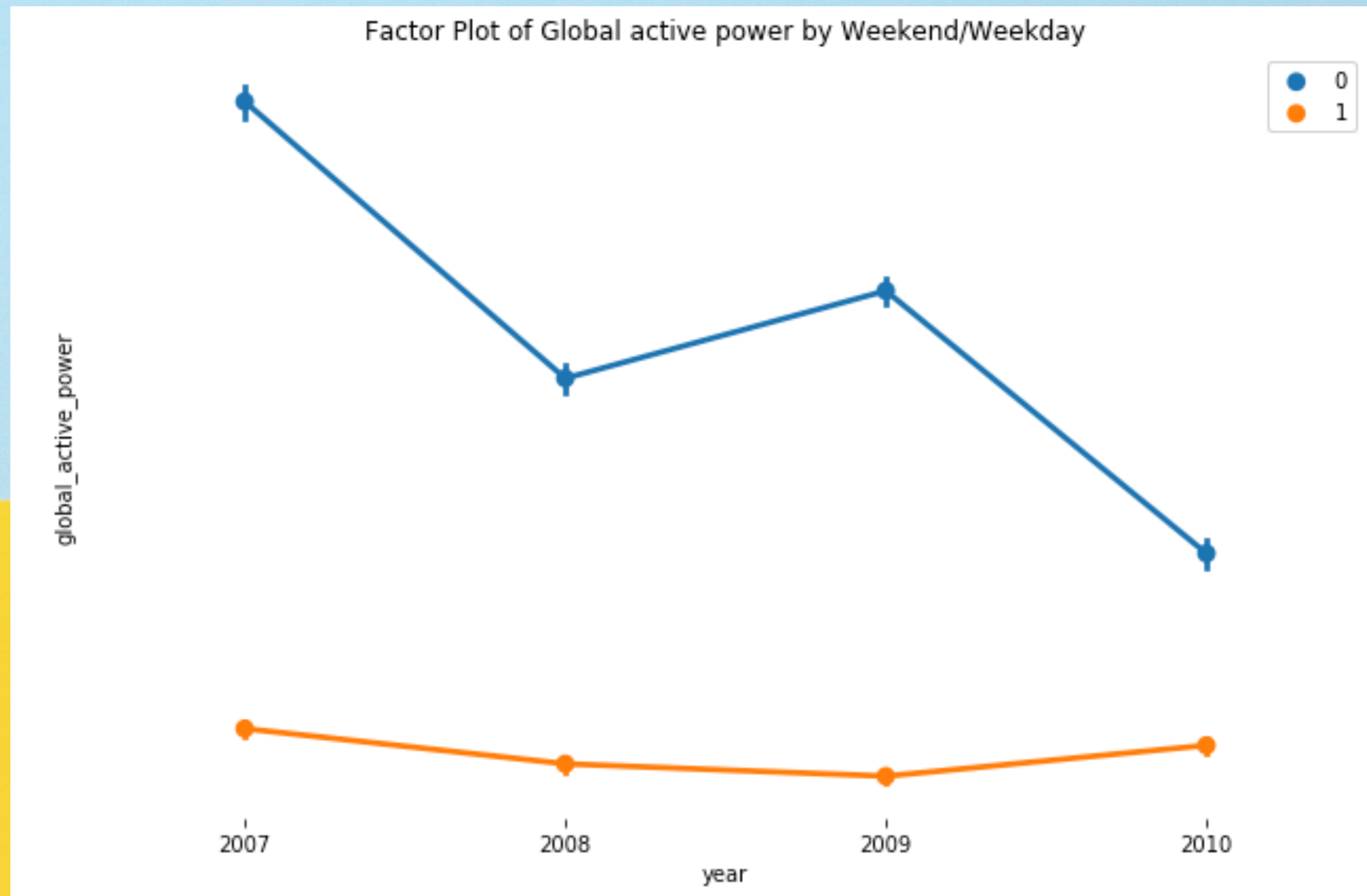


# More energy was consumed on weekends than during weekdays over the four years

On average, more energy is consumed on the weekend (0) than on weekdays (1)

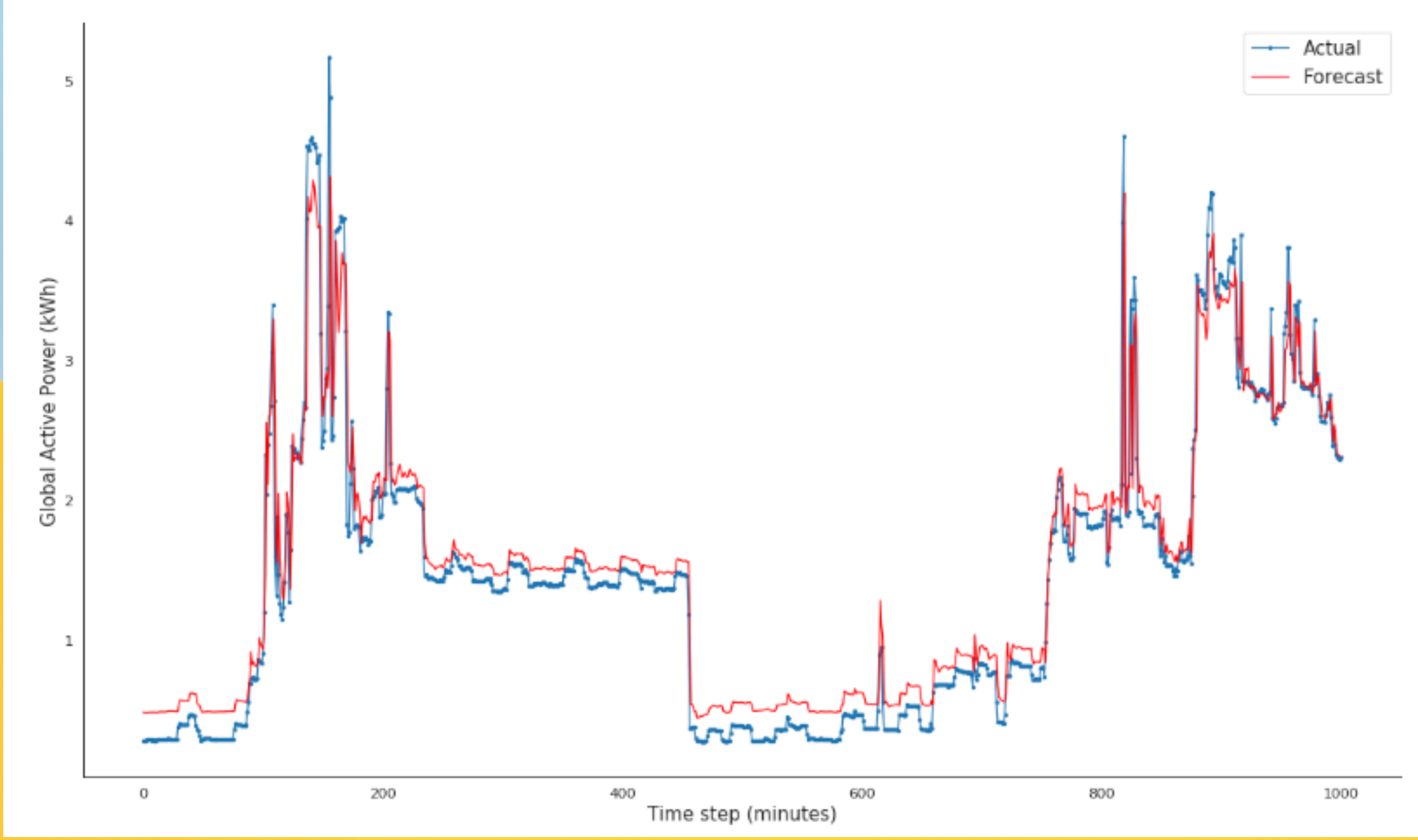
People have commitments such as school or work

But most likely spend more time at home on the weekends



# Neural Network Modelling provided the best forecasts

Using this forecasting technique proved to be the most effective for long-term predictions



# A solution to the problem

Data-driven answer which successfully incorporates seasonal factors

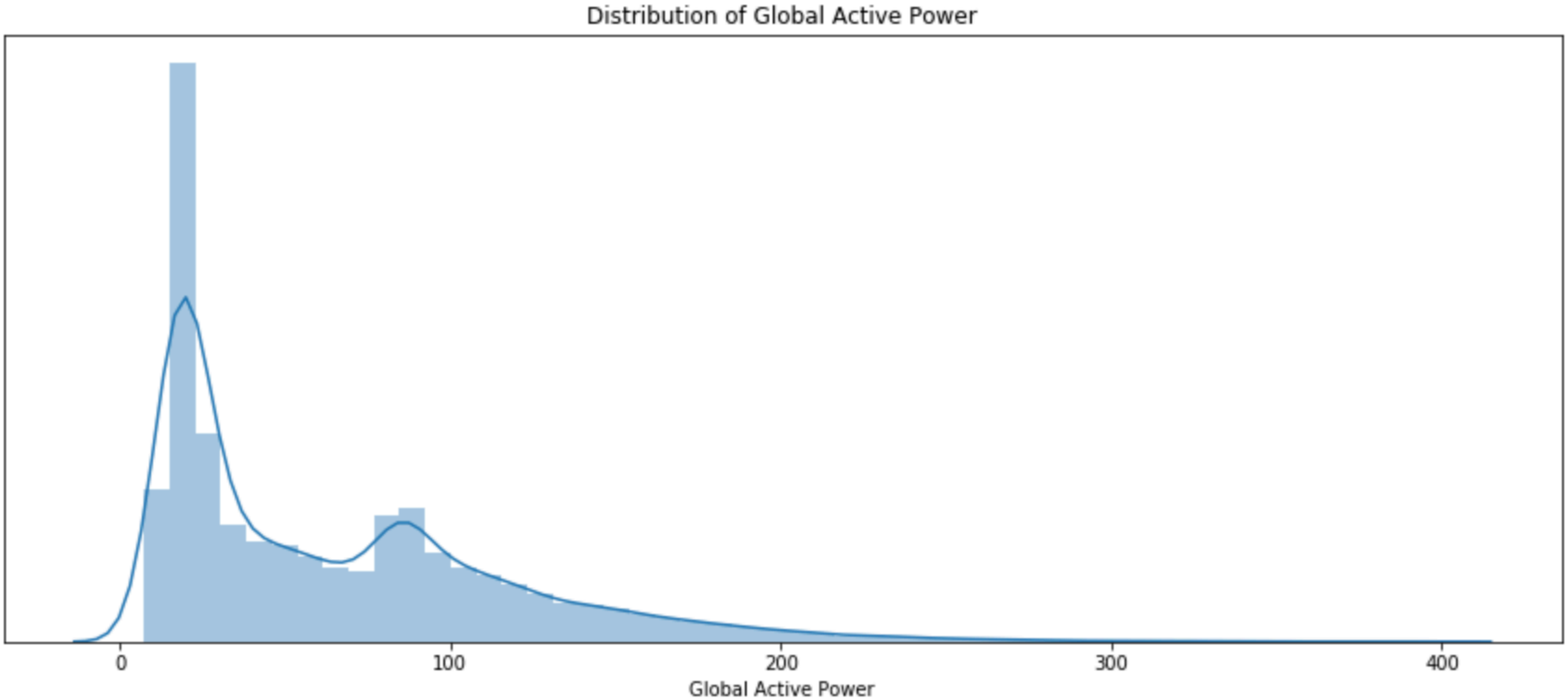
Using a dynamic implementation of energy supply using an LSTM model can identify energy demand

**Increase profitability and environmental welfare**


Thank you!

# APPENDIX

# Distribution of Global Active Power



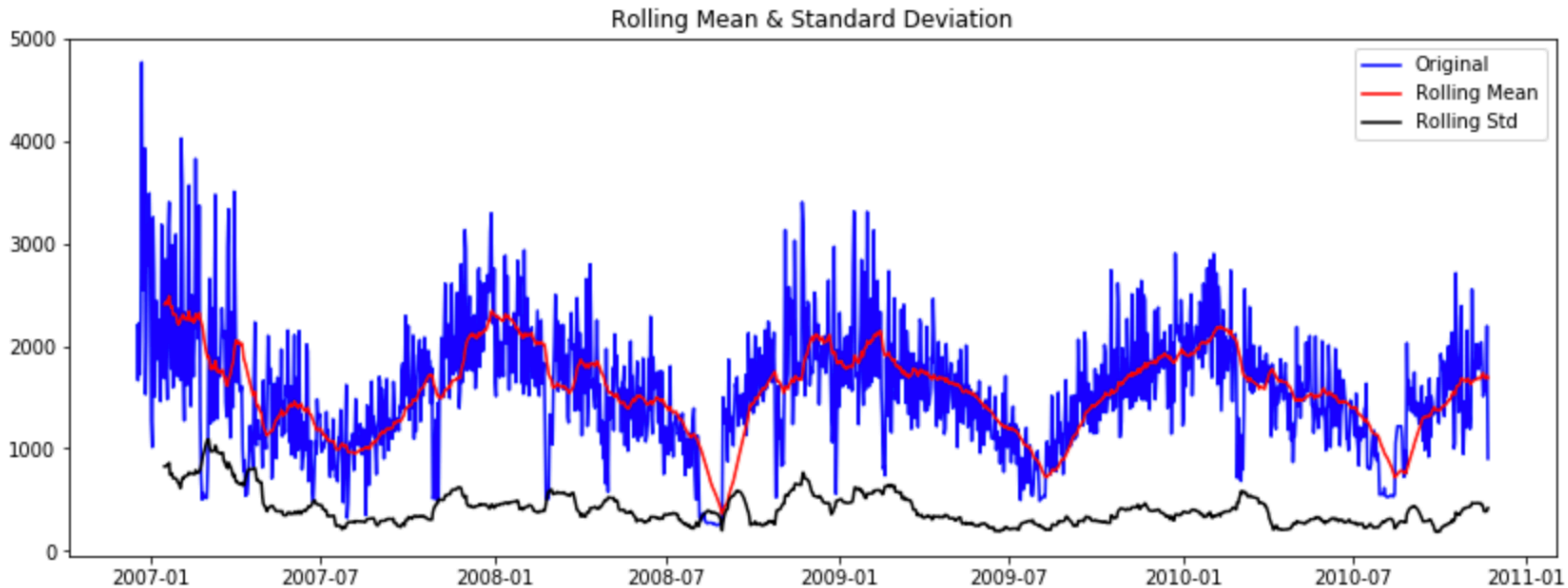

```
Kurtosis of Global Active Power Distribution: 1.8002245114936706
Skewness of Global Active Power Distribution: 1.3060675051853323
```

There appears to be two peaks in global active power, suggesting a *bimodal* distribution

# Dickey-Fuller Test (adjusted for daily totals)

```python
# Resampling over day totals to reduce noise
data_days = data.resample('D')
data_days = data_days.sum()

test_stationarity(data_days.global_active_power)
```




```
<Results of Dickey-Fuller Test>
Test Statistic                 -3.666034
p-value                         0.004617
#Lags Used                     22.000000
Number of Observations Used  1412.000000
Critical Value (1%)            -3.434990
Critical Value (5%)            -2.863589
Critical Value (10%)           -2.567861
```

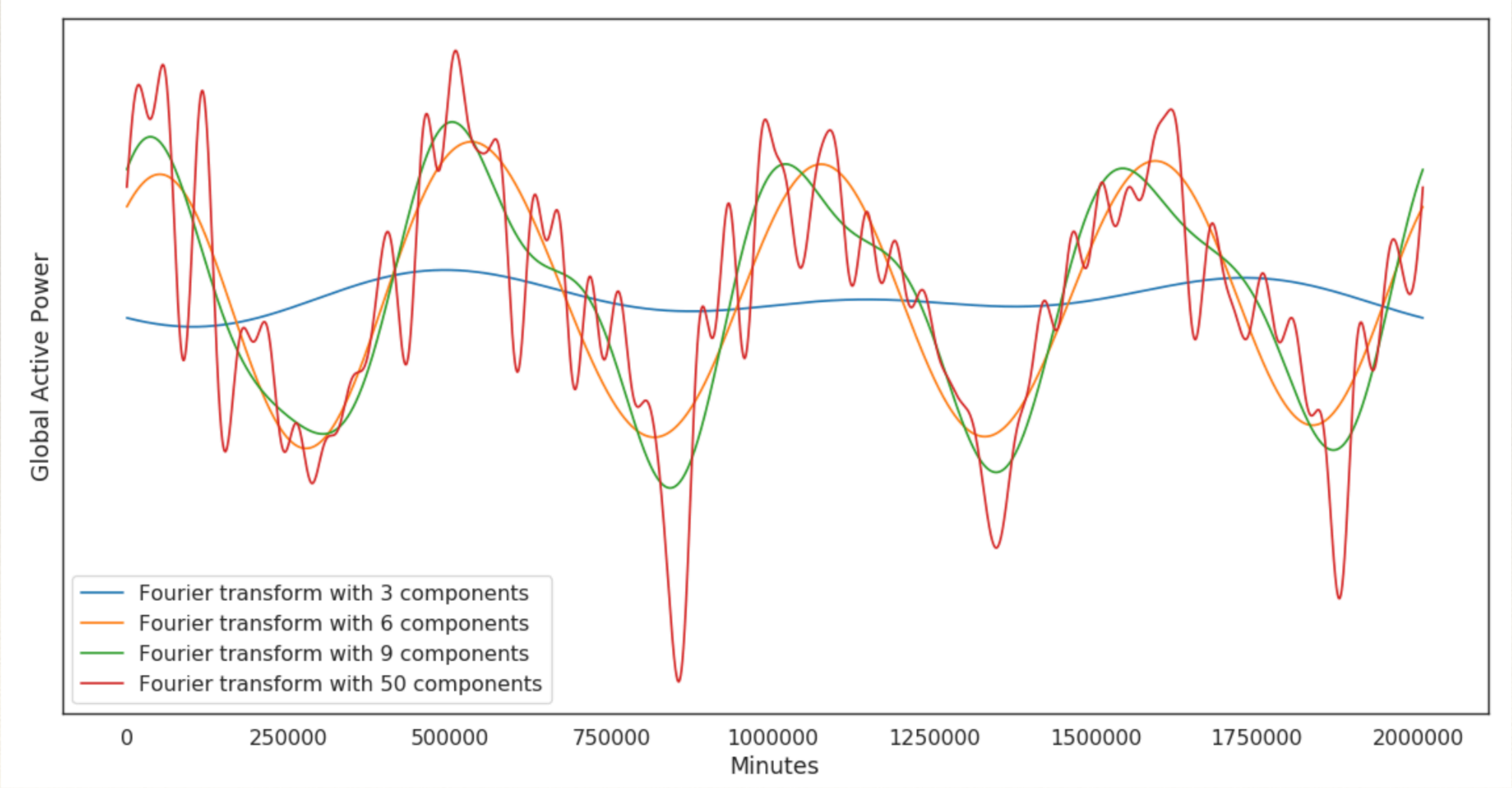
p-value < 0.05 - Reject the null hypothesis that the time series has a ‘unit root’. Over time patterns are similar enough to be deemed stationary

# Fourier Transformations

```
global_active_power_fft = np.fft.fft(np.asarray(df.global_active_power.tolist()))
fft_df = pd.DataFrame({'fft':global_active_power_fft})
fft_df['absolute'] = fft_df['fft'].apply(lambda x: np.abs(x))
fft_df['angle'] = fft_df['fft'].apply(lambda x: np.angle(x))
```

```
plt.figure(figsize=(14, 7), dpi=100)
fft_list = np.asarray(fft_df['fft'].tolist())
for i in [3, 6, 9, 50]:
    fft_list_m10 = np.copy(fft_list); fft_list_m10[i:-i]=0
    plt.plot(np.fft.ifft(fft_list_m10), label='Fourier transform with {} components'.format(i))
plt.xlabel('Minutes')
plt.ylabel('Global Active Power')
plt.yticks([])
plt.legend()
plt.show();
```

For data that has seasonal, or daily patterns, we can use Fourier analysis trends and make predictions. It is a combination of extrapolation and denoising



# Naive Model Results


```
daily: [46.279]
```

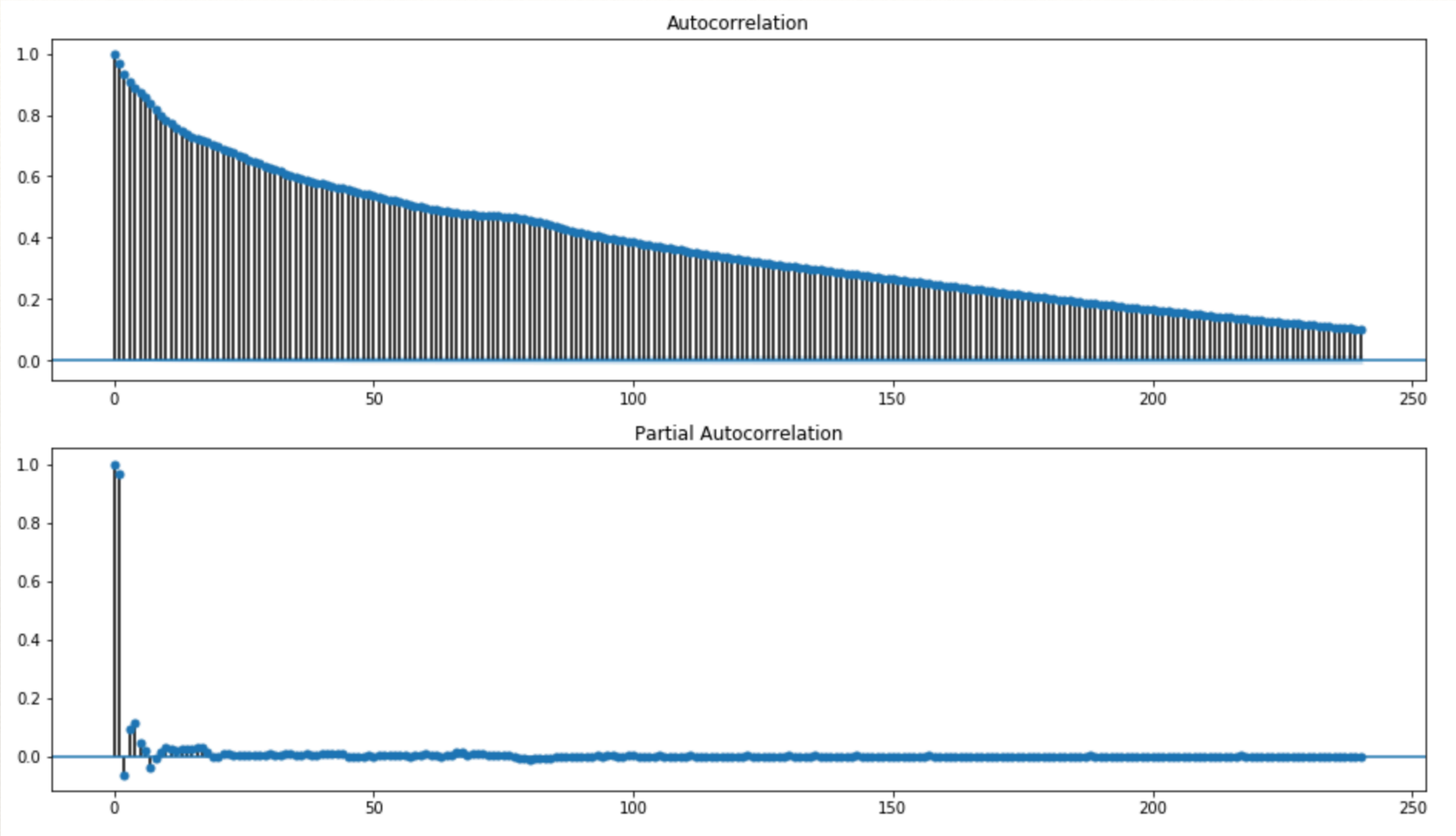
```
weekly: [48.971]
```

```
week-oya: [48.610]
```

Hourly predictions in RMSE (root mean squared error)
Daily forecast has the lowest error

Daily, weekly and weekly one-year-average predictions.
Predictions are global active power hourly consumption.

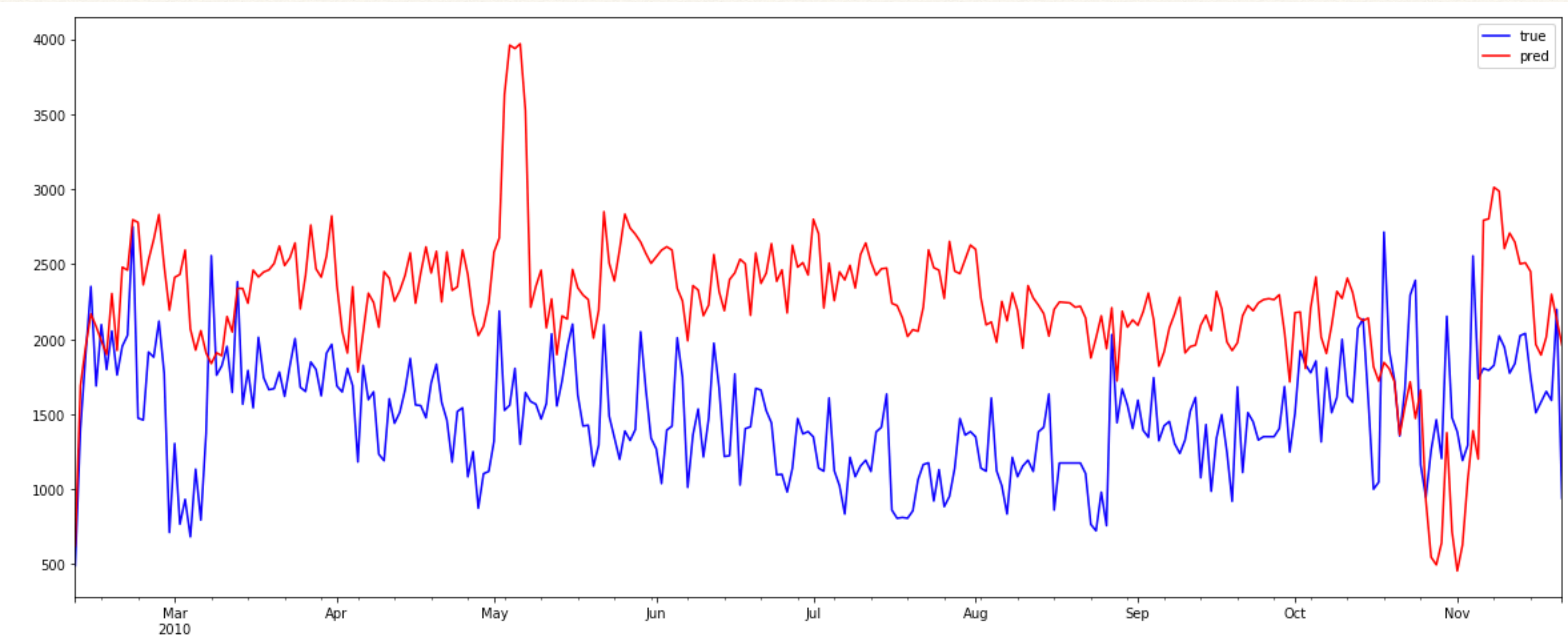
# Autocorrelation & Partial Autocorrelation




Lags are consistent with thinking. Previous 240 lags (i.e. 3 hours) are all significantly correlated at time *t* which is shown by autocorrelation

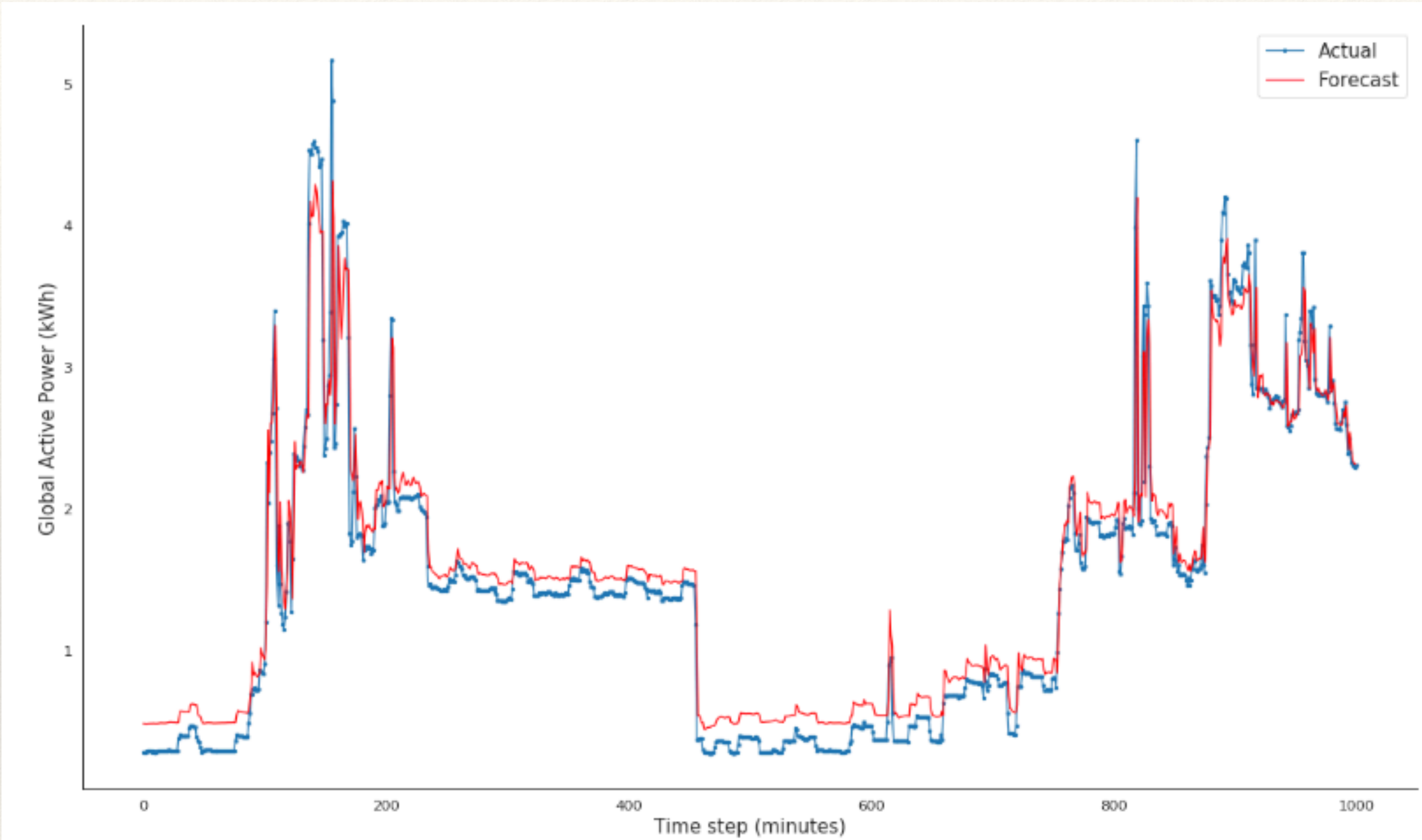
Partial autocorrelation shows that only a few lags are correlated with values at time *t*

# Autoregressive Moving Average Model




**Hourly predictions with  average RMSE of 51.5**

# LSTM Model




```
Train Mean Absolute Error: 0.20987012931663987
Train Root Mean Squared Error: 0.30189010249364967
Test Mean Absolute Error: 0.189517954561053
Test Root Mean Squared Error: 0.2597281292076344
```

mean squared error)
Scaled to hourly, RMSE on test set of ~ 15
Performance considerably better than Naive or SARIMAX model