Time Series Forecasting with a SARIMA Model

Predicting daily electricity loads for a building on the UC Berkeley campus

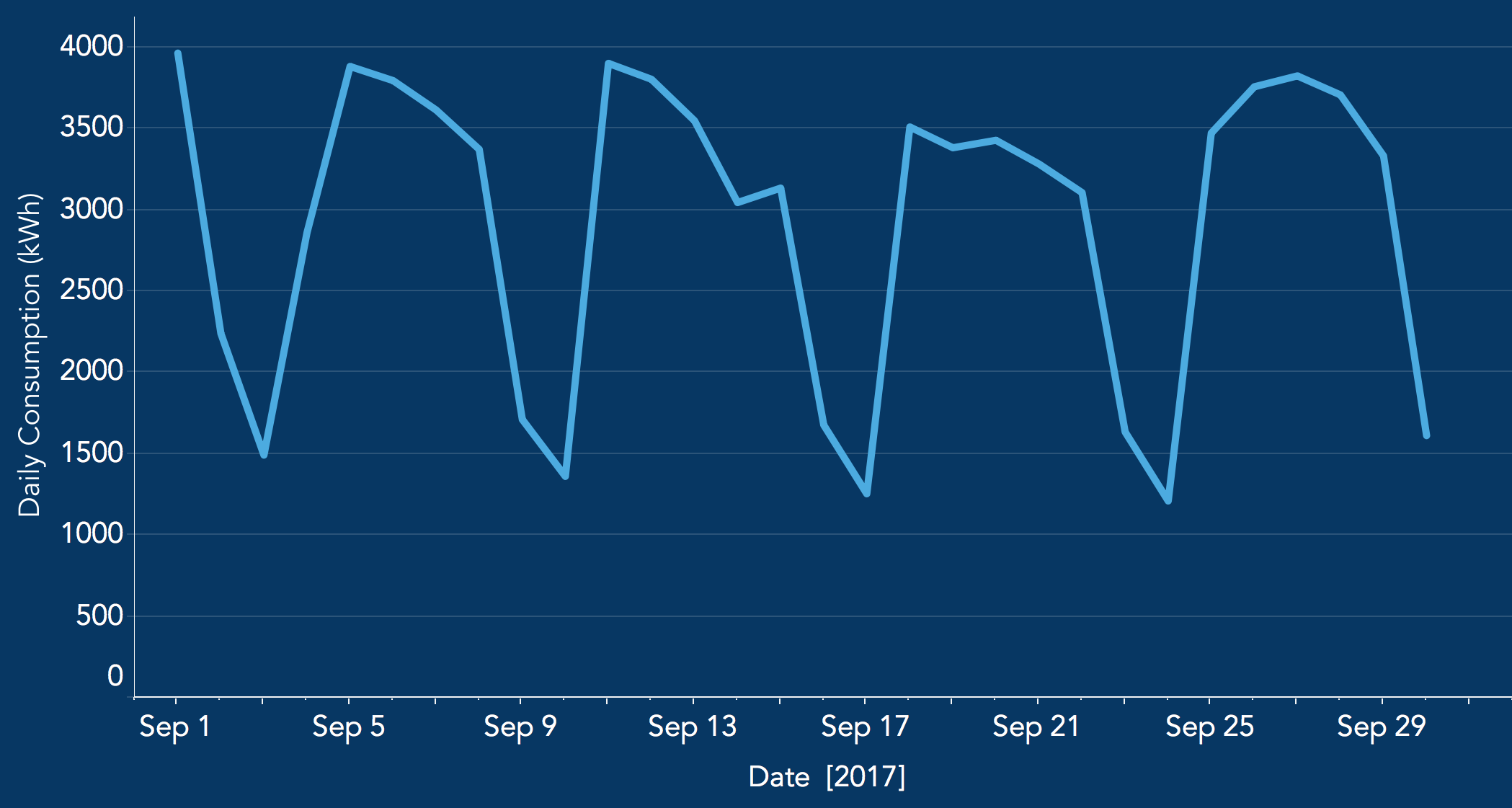
<https://towardsdatascience.com/time-series-forecasting-with-a-sarima-model-db051b7ae459>

In this an example of electricity load forecasting using a SARIMA model. Three years of [daily electricity load data](https://engagementdashboard.com/ucb/ucb/)was gathered for a building on the UC Berkeley campus to create a model based on the building’s electricity use from January 2016 through October 2019. The model will predict the daily loads for November 2019, and its effectiveness will be measured using mean absolute percentage error. The following topics will be covered while detailing this process:

1. **Stationarity (Differencing and Augmented Dickey-Fuller Test)**
2. **ACF and PACF Plots**
3. **Grid Search and AIC**
4. **Walk Forward Validation**
5. **MAPE**
6. **Exogenous Variables**

**EDA and Stationarity Check**

Before modeling, let’s take a look at some data to see if we can extract any meaningful information.

Figure 1: Daily Electricity Load Example

Looking at Figure 1 above, there is a very clear weekly trend in the data. While there is a high electricity load when class is in session during the week, there is a lower electricity load during the weekend. This piece of insight will come in handy later on when determining the SARIMA parameters.

For many types of time series models, it’s important to verify that your data is stationary. As a quick summary, the data needs to satisfy the following requirements to ensure stationarity:

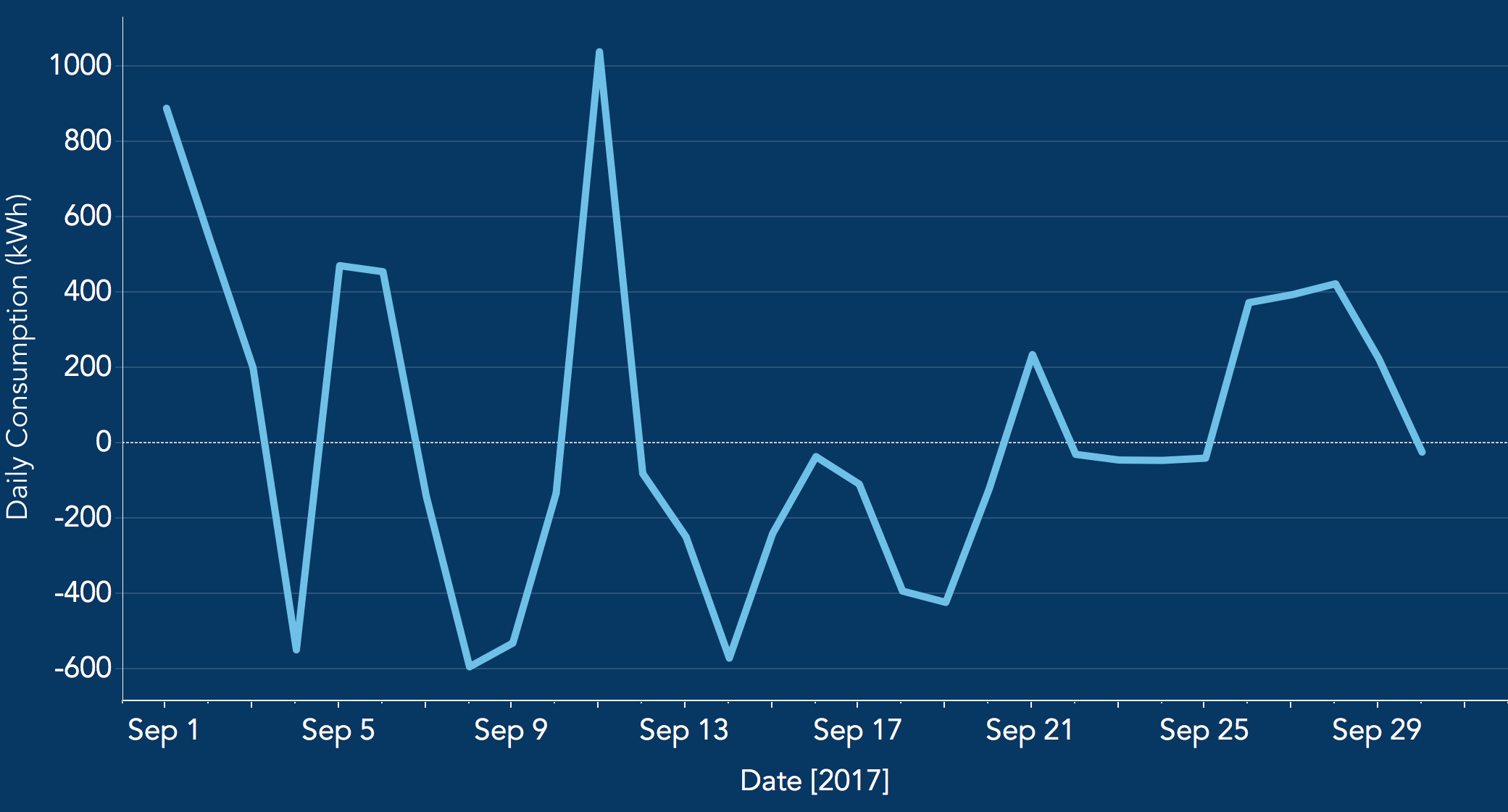
Constant mean

Constant variance

Constant autocorrelation structure (current value being dependent on past value)

No periodic component

It’s clear from the EDA above that the data contains a periodic component, so transformations will need to be applied to make the data stationary. In this case, we’ll apply both a first difference and weekly seasonal difference. Figure 2 below shows the same date range after the data has been differenced; it no longer contains the weekly trend.

Figure 2: Daily Load After First Difference and Seasonal Difference

A quick and easy way to verify the data’s stationarity is through the augmented Dickey-Fuller test. This can be done with the following code:

from statsmodels.tsa.stattools import adfulleradf, pvalue, usedlag\_, nobs\_, critical\_values\_, icbest\_ = adfuller(train)

The null hypothesis (p-value > 0.05) for this test is that the data is not stationary. Based on our output from the code above, the p-value was calculated to be ~ 0, so we can reject the null hypothesis and say that the data is in fact stationary. On to modeling!

SARIMA Model Parameters — ACF and PACF Plots

As a quick overview, [SARIMA models](http://people.duke.edu/~rnau/Slides_on_ARIMA_models--Robert_Nau.pdf) are ARIMA models with a seasonal component. Per the formula SARIMA(*p*,*d*,*q*)x(*P*,*D*,*Q,s*), the parameters for these types of models are as follows:

*p* and seasonal *P*: indicate number of autoregressive terms (lags of the stationarized series)

*d* and seasonal *D*: indicate differencing that must be done to stationarize series

*q* and seasonal *Q*: indicate number of moving average terms (lags of the forecast errors)

*s*: indicates seasonal length in the data

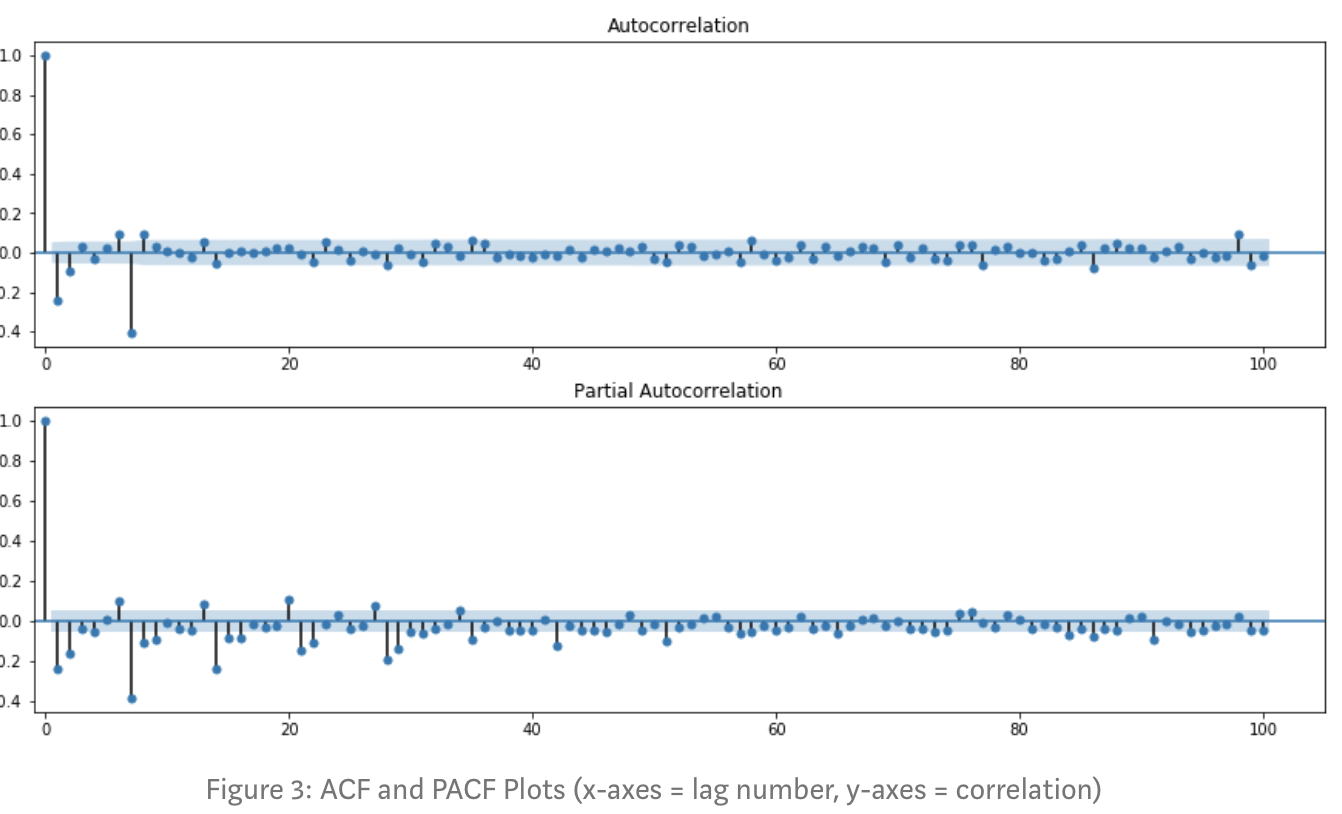
Two approaches were taken to determine the ideal SARIMA parameters: [ACF and PACF plots](http://people.duke.edu/~rnau/411arim3.htm), and a grid search. The ACF and PACF plots were used as a starting point to narrow down to a few potential parameters, and then a grid search was used to identify the best parameters. Let’s take a look!

ACF and PACF plots were generated using the following code, and can be seen in Figure 3 below.

import matplotlib.pyplot as plt  
#import statsmodels.api as sm sm.tsa.statespace

from statsmodels.api.tsa.statespace import SARIMAX

fig, ax = plt.subplots(2,1)  
fig = sm.graphics.tsa.plot\_acf(train, lags=50, ax=ax[0])  
fig = sm.graphics.tsa.plot\_pacf(train, lags=50, ax=ax[1])  
plt.show()



As can be seen, there are significant negative spikes in the ACF plot at lag 1 and lag 7 (weekly seasonal component). In the PACF plot, there are significant negative spikes at lags 1 and 2 (i.e. lags are slower to decay), as well as significant negative spikes at lags 7, 14, 21, etc. These characteristics suggest a potential parameter starting point of **SARIMA(0,1,1)x(0,1,1,7):**

*p* and *P* both equal 0 (no significant positive spikes in ACF and PACF plots)

*d* and *D* both equal 1 (first difference and seasonal difference)

*q* and *Q* both equal 1 (significant negative spikes, PACF decay is more gradual)

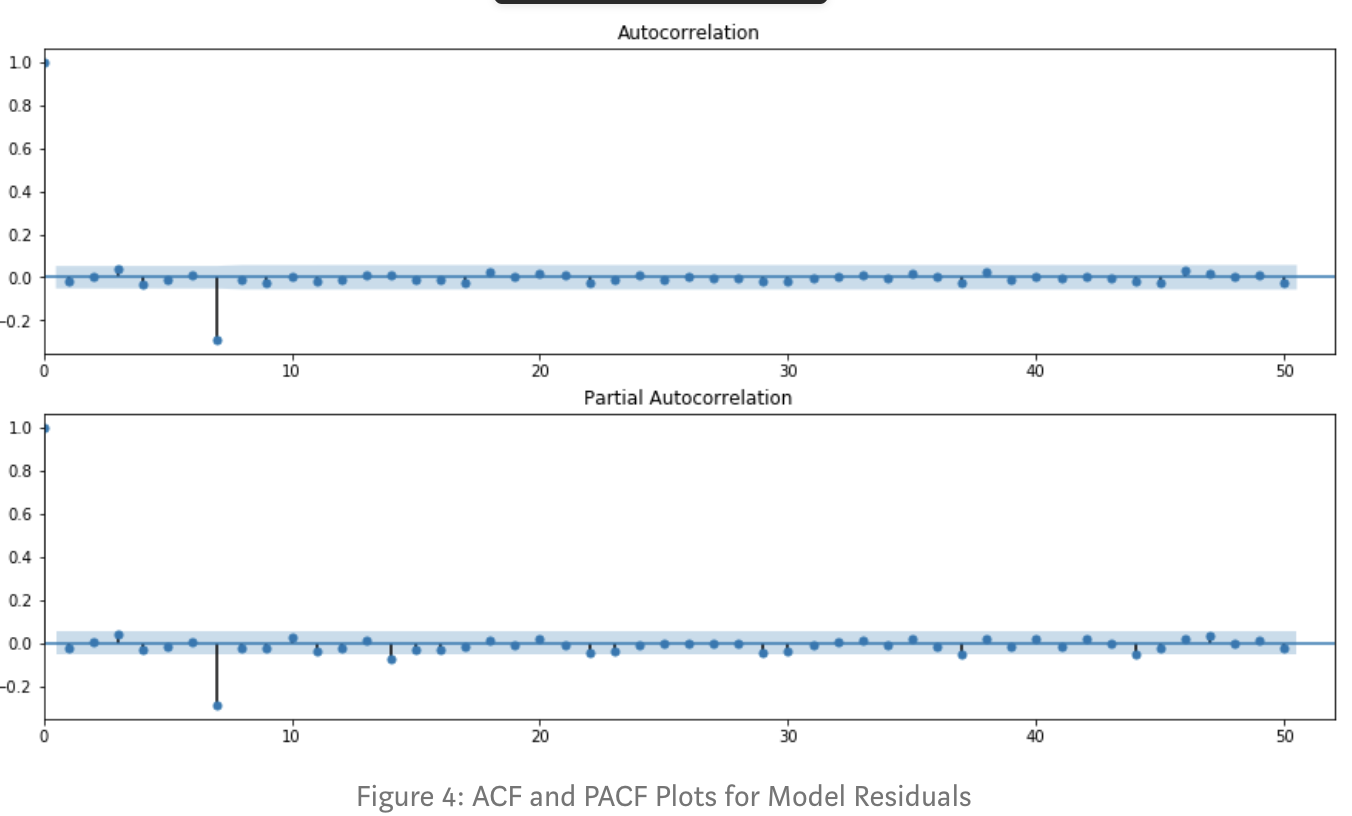
*s* equal to 7 (weekly seasonal component)

Using these parameters, the model was fit to the data, and then a new set of ACF and PACF plots of the model’s residuals was evaluated. See the code below:

model = SARIMAX(train, order=(0,1,1), seasonal\_order(0,1,1,7))  
 .fit(max\_iter=50, method='powell')

res = model.residfig, ax = plt.subplots(2,1)  
fig = sm.graphics.tsa.plot\_acf(res, lags=50, ax=ax[0])  
fig = sm.graphics.tsa.plot\_pacf(res, lags=50, ax=ax[1])  
plt.show()

See Figure 4 below for the ACF and PACF plots of the residuals. There are no significant spikes except for at lag 7. These parameters are an acceptable starting point for the model and can be further improved using a grid search.



# SARIMA Model Parameters — Grid Search

With a starting point of (0,1,1)x(0,1,1,7), a grid search was set up to test a number of different parameter combinations. It’s worth looking at models where *p* = 0,1, or 2 and *q* = 1 or 2. A metric that can be used to compare the relative quality of each model is [AIC](https://en.wikipedia.org/wiki/Akaike_information_criterion) (Akaike information criterion), which is an estimator of out-of-sample prediction error; a lower AIC score indicates a more predictive model. See the code below that generated the list of different parameter combinations, as well as the SARIMA model pipeline that tested each combination and printed the corresponding AIC scores:

import itertools

#set parameter range  
p = range(0,3)  
q = range(1,3)  
d = range(1,2)  
s = range(7,8)# list of all parameter combos  
pdq = list(itertools.product(p, d, q))  
seasonal\_pdq = list(itertools.product(p, d, q, s))# SARIMA model pipeline  
for param in pdq:  
 for param\_seasonal in seasonal\_pdq:  
 try:  
 mod = SARIMAX(train, order=param, seasonal\_order=param\_seasonal)  
 results = mod.fit(max\_iter = 50, method = 'powell')

print('SARIMA{},{} - AIC:{}'.format(param, param\_seasonal, results.aic))  
 except:  
 continue

In this case, the lowest AIC score was achieved with *p*, *P* = 1 and *q*, *Q* = 2, so the final SARIMA model parameters to be used are **SARIMA(1,1,2)x(1,1,2,7)**.

# Walk Forward Validation & Evaluation with MAPE

As mentioned in the background of this article, the SARIMA model is predicting the daily electricity load for November 2019. While a simple train/test split is possible for time series (where the test data is the actual November 2019 load data in our case), a more robust option is [walk forward validation](https://www.liip.ch/en/blog/time-series-prediction-a-short-comparison-of-best-practices) in which the test data is split up multiple times. See code below; in the first walk through, daily loads for the first week of November are predicted based on the model fit to the training data. In the second walk through, the actual daily loads for the first week of November are added to the training data, the model is refit, and loads for the second week of November are predicted. The same process is followed for the third and fourth weeks of November.

Mean absolute percentage error (MAPE) is a metric that can be used to evaluate the SARIMA model. For each predicted data point, the absolute difference from the corresponding test point was calculated, and then divided by the test point. The average percentage gives the MAPE. See the code below for walk forward validation and MAPE calculation.

# data = entire dataset  
# n\_test = point where data is split into training and test sets

def **walk\_forward\_validation**(data, n\_test):

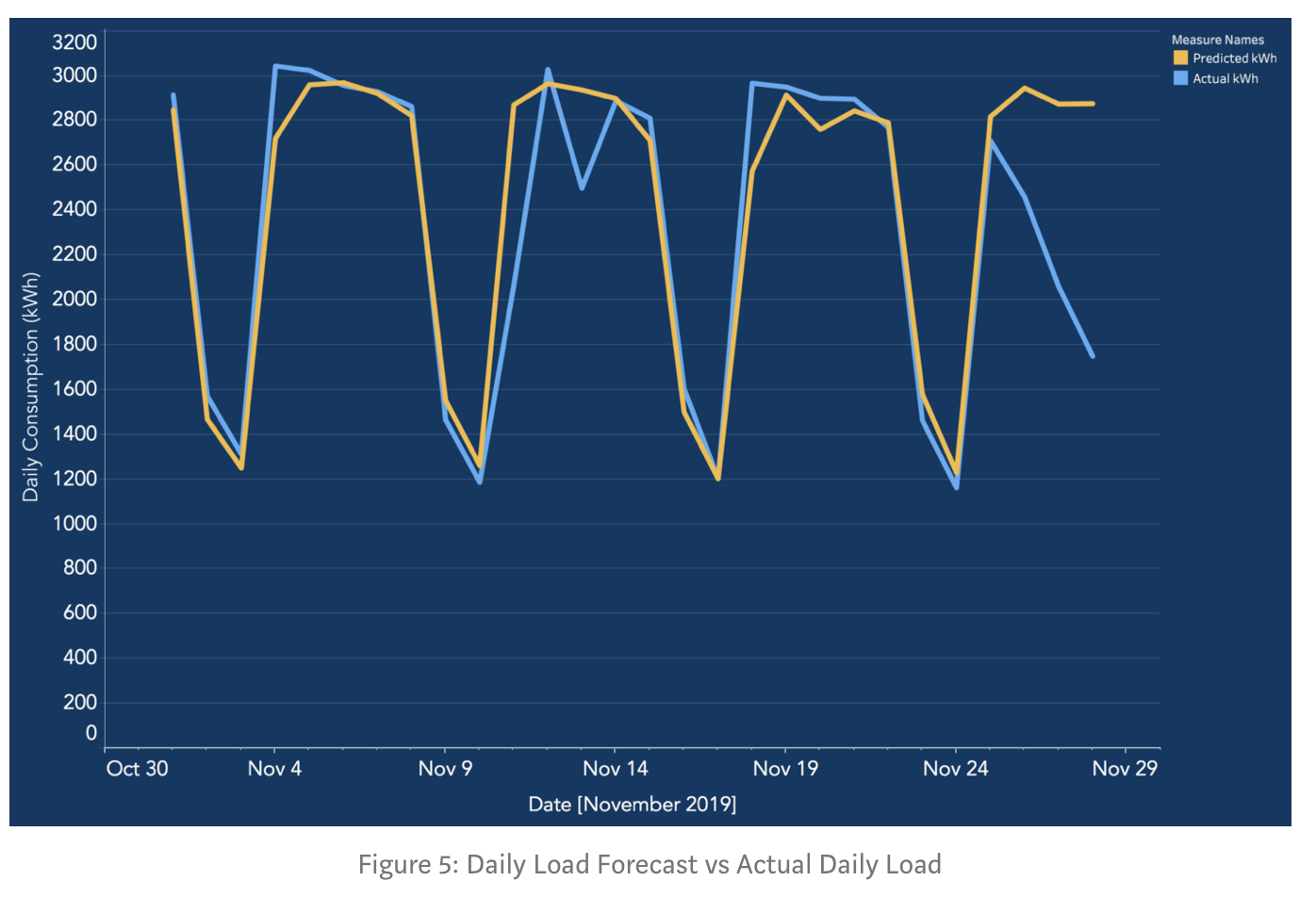
predictions = np.array([])  
 mape\_list = []  
 train, test = data[:n\_test], data[n\_test:]  
 day\_list = [7,14,21,28] # weeks 1,2,3,4

for i in day\_list:  
 # Fit model to training data  
 model = sm.tsa.statespace.SARIMAX(train, order=(1,1,2), seasonal\_order(1,1,2,7))   
 .fit(max\_iter = 50, method = 'powell')  
   
 # Forecast daily loads for week i  
 forecast = model.get\_forecast(steps = 7)  
 predictions = np.concatenate(predictions, forecast, axis=None)

# Calculate MAPE and add to mape\_list  
 j = i-7  
 mape\_score = (abs(test[j:i]-predictions[j:i])/test[j:i])\*100  
 mape\_mean = mape\_score.mean()  
 mape\_list.append(mape\_mean) # Add week i to training data for next loop  
 train = np.concatenate((train, test[j:i]), axis=None)

return predictions, mape\_list

Our model’s predictions had a MAPE of **9.74%**. Not bad! Let’s plot the predictions along with the actual data to see how they compare visually. See Figure 5 below.



The predictions seem to follow the weekly trend fairly well, although there’s pretty significant error for a number of the days throughout the month. Taking a closer look, these are holidays (specifically Veteran’s Day on the 11th, and Thanksgiving Break at the end of the month).

**Exogenous Variables**

A quick way to improve the model’s performance is to add in an exogenous variable that accounts for holidays throughout the year. Python has a “holidays” package that can be imported and used to determine whether or not a date is a holiday (U.S. holidays can be specified).

The SARIMAX model that we’ve been using so far simply takes in exogenous variables as an additional argument. See the code excerpt below:

model2 = SARIMAX(train, holiday\_ex\_var, order=(1,1,2), seasonal\_order(1,1,2,7))  
.fit(max\_iter = 50, method = 'powell')

Running this model through the walk\_forward\_validation function, we get a MAPE of 7.04%. This is more than a 25% improvement on the initial model! See Figure 6 below for the updated forecast in comparison with the actual daily loads. There is a noticeable improvement for the Veteran’s Day and Thanksgiving load predictions; adding in the holiday variable was beneficial! There is still room for improvement however, as there is still significant error for other days during Thanksgiving break. Methods to reduce those errors will not be explored here.