MEASURE ENERGY CONSUMPTION

Phase 3: Development Part_3

Energy Consumption

Intoduction:

To better follow the energy consumption, the government wants energy suppliers to install smart meters in every home in England, Wales and Scotland. There are more than 26 million homes for the energy suppliers to get to, with the goal of every home having a smart meter by 2020.
This roll out of meter is lead by the European Union who asked all member governments to look at smart meters as part of measures to upgrade our energy supply and tackle climate change. After an initial study, the British government decided to adopt smart meters as part of their plan to update our ageing energy system.

In this dataset, you will find a refactorized version of the data from the London data store, that contains the energy consumption readings for a sample of 5,567 London Households that took part in the UK Power Networks led Low Carbon London project between November 2011 and February 2014. The data from the smart meters seems associated only with electrical consumption.

Daily Energy Data Preparation

Importing Libraries

```
#!pip install pmdarima

import pandas as pd
import numpy as np
from pandas import datetime
from matplotlib import pyplot as plt
import os
from statsmodels.tsa.arima_model import ARIMA
from matplotlib import pyplot
from pandas.tools.plotting import autocorrelation_plot

#from pyramid.arima import auto_arima
#from pmdarima.arima import auto_arima
import pyflux as pf
from sklearn.cluster import KMeans
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
import statsmodels.api as sm
from statsmodels.tsa.statespace.sarimax import SARIMAX

import math

from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error

from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
```

Energy Data

We are predicting for energy demand in the future- therefore we are taking only energy sum i.e. total energy use per day for a given household.

```
# Combining all blocks
for num in range(0,112):
    df =
pd.read_csv("../input/daily_dataset/daily_dataset/block_"+str(num)+".csv")
    df = df[['day','LCLid','energy_sum']]
    df.reset_index()
    df.to_csv("hc_"+str(num)+".csv")

fout= open("energy.csv","a")
# first file:
for line in open("hc_0.csv"):
    fout.write(line)
# now the rest:
for num in range(0,112):
    f = open("hc_"+str(num)+".csv")
    f.readline() # skip the header
    for line in f:
         fout.write(line)
    f.close()
```

```
fout.close()
```

Energy at Day Level

```
energy = pd.read_csv('energy.csv')
len(energy)
```

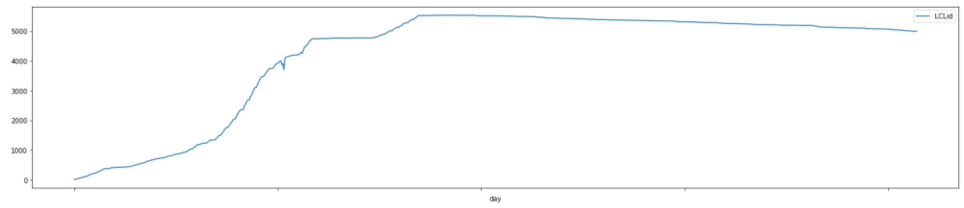
```
3536007
```

House Count

In the dataset we see that the number of households for which energy data was collected across different days are different. This is probably due to the gradually increasing adoption of smart meters in London. This could lead to false interpretation that the energy for a particular day might be high when it could be that the data was only collected for more number of houses. We will look at the house count for each day.

```
housecount = energy.groupby('day')[['LCLid']].nunique()
housecount.head(4)housecount.plot(figsize=(25,5))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1d0e44d6d8>
```



Normalization across households

The data collection across households are inconsistent- therefore we will be using *energy per household* as the target to predict rather than energy alone. This is an optional step as we can also predict for energy sum as whole for each household. However there are quite a lot of unique households for which we have to repeat the exercise and our ultimate goal is to predict overall consumption forecast and not at household level.
This also means that since household level is removed, we are not looking into the ACORN details which is available at household level

```
energy = energy.groupby('day')[['energy_sum']].sum()
energy = energy.merge(housecount, on = ['day'])
energy = energy.reset_index()
```

```
energy.count()
```

```
day           829
energy_sum    829
LCLid         829
dtype: int64
```

```
energy.day = pd.to_datetime(energy.day,format='%Y-%m-%d').dt.date
```

```
energy['avg_energy'] =  energy['energy_sum']/energy['LCLid']
print("Starting Point of Data at Day Level",min(energy.day))
print("Ending Point of Data at Day Level",max(energy.day))
```

```
Starting Point of Data at Day Level 2011-11-23
Ending Point of Data at Day Level 2014-02-28
```

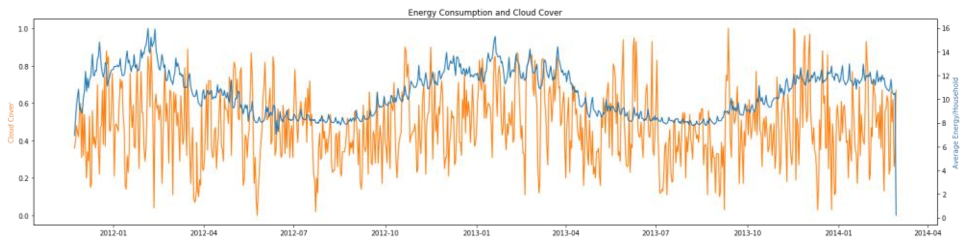
```
energy.describe()
```

	energy_sum	LCLid	avg_energy
count	829.000000	829.000000	829.000000
mean	43535.325676	4234.539204	10.491862
std	20550.594031	1789.994799	1.902513
min	90.385000	13.000000	0.211766
25%	34665.436003	4084.000000	8.676955
50%	46641.160997	5138.000000	10.516983
75%	59755.616996	5369.000000	12.000690
max	84156.135002	5541.000000	15.964434

3. Cloud Cover

The cloud cover value seems to be following the same pattern as the energy consumption.

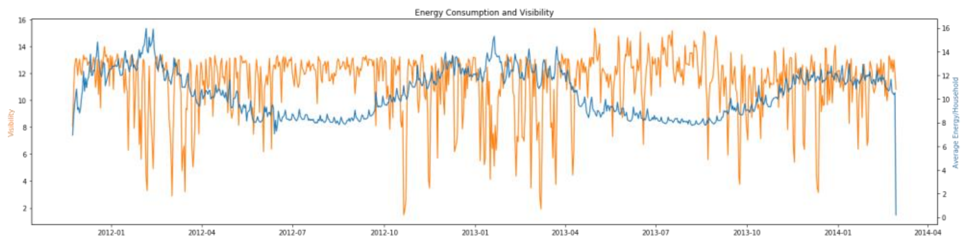
```
fig, ax1 = plt.subplots(figsize = (20,5))
ax1.plot(weather_energy.day, weather_energy.cloudCover, color =
'tab:orange')
ax1.set_ylabel('Cloud Cover',color = 'tab:orange')
ax2 = ax1.twinx()
ax2.plot(weather_energy.day,weather_energy.avg_energy,color = 'tab:blue')
ax2.set_ylabel('Average Energy/Household',color = 'tab:blue')
plt.title('Energy Consumption and Cloud Cover')
fig.tight_layout()
plt.show()
```



4. Visibility

The visibility factor does not seem to affect energy consumption at all- since visibility is most likely an outdoors factor, it is unlikely that it's increase or decrease affects energy consumption within a household.

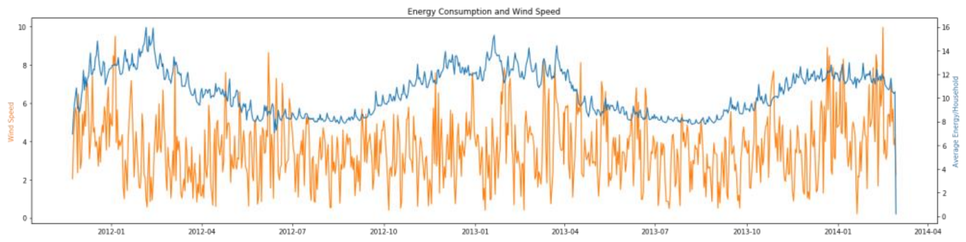
```
fig, ax1 = plt.subplots(figsize = (20,5))
ax1.plot(weather_energy.day, weather_energy.visibility, color =
'tab:orange')
ax1.set_ylabel('Visibility',color = 'tab:orange')
ax2 = ax1.twinx()
ax2.plot(weather_energy.day,weather_energy.avg_energy,color = 'tab:blue')
ax2.set_ylabel('Average Energy/Household',color = 'tab:blue')
plt.title('Energy Consumption and Visibility')
fig.tight_layout()
plt.show()
```



5. Wind Speed

Like visibility, wind speed seems to be an outdoors factor which does not affect in the energy consumption as such.

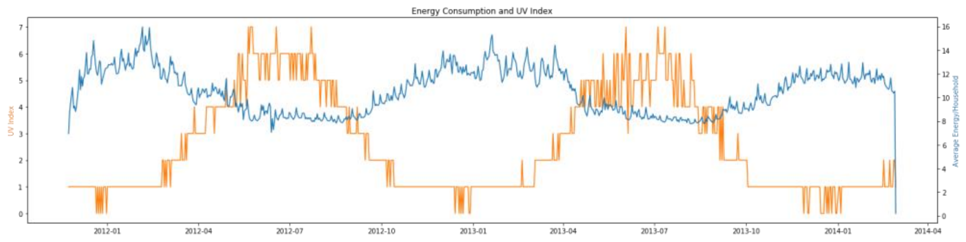
```
fig, ax1 = plt.subplots(figsize = (20,5))
ax1.plot(weather_energy.day, weather_energy.windSpeed, color =
'tab:orange')
ax1.set_ylabel('Wind Speed',color = 'tab:orange')
ax2 = ax1.twinx()
ax2.plot(weather_energy.day,weather_energy.avg_energy,color = 'tab:blue')
ax2.set_ylabel('Average Energy/Household',color = 'tab:blue')
plt.title('Energy Consumption and Wind Speed')
fig.tight_layout()
plt.show()
```



6. UV Index

The UV index has an inverse relationship with energy consumption- why?

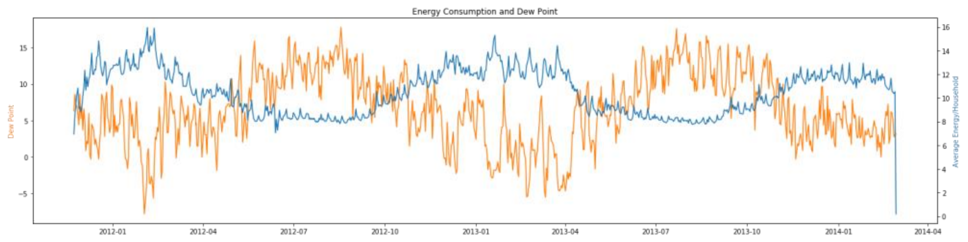
```
fig, ax1 = plt.subplots(figsize = (20,5))
ax1.plot(weather_energy.day, weather_energy.uvIndex, color = 'tab:orange')
ax1.set_ylabel('UV Index',color = 'tab:orange')
ax2 = ax1.twinx()
ax2.plot(weather_energy.day,weather_energy.avg_energy,color = 'tab:blue')
ax2.set_ylabel('Average Energy/Household',color = 'tab:blue')
plt.title('Energy Consumption and UV Index')
fig.tight_layout()
plt.show()
```



7. dewPoint

Dew Point- is a function of humidity and temperature therefore it displays similar relation to energy consumption.

```
fig, ax1 = plt.subplots(figsize = (20,5))
ax1.plot(weather_energy.day, weather_energy.dewPoint, color =
'tab:orange')
ax1.set_ylabel('Dew Point',color = 'tab:orange')
ax2 = ax1.twinx()
ax2.plot(weather_energy.day,weather_energy.avg_energy,color = 'tab:blue')
ax2.set_ylabel('Average Energy/Household',color = 'tab:blue')
plt.title('Energy Consumption and Dew Point')
fig.tight_layout()
plt.show()
```



Correlation between Weather Variables and Energy Consumption

- Energy has high positive correlation with humidity and high negative correlation with temperature.
- Dew Point, UV Index display multicollinearity with Temperature, hence discarded
- Cloud Cover and Visibility display multicollinearity with Humidity, hence discarded
- Pressure and Moon Phase have minimal correlation with Energy, hence discarded
- Wind Speed has low correlation with energy but does not show multicollinearity

```
cor_matrix = weather_energy[['avg_energy','temperatureMax','dewPoint',
'cloudCover', 'windSpeed','pressure', 'visibility', 'humidity','uvIndex',
'moonPhase']].corr()
cor_matrix
```

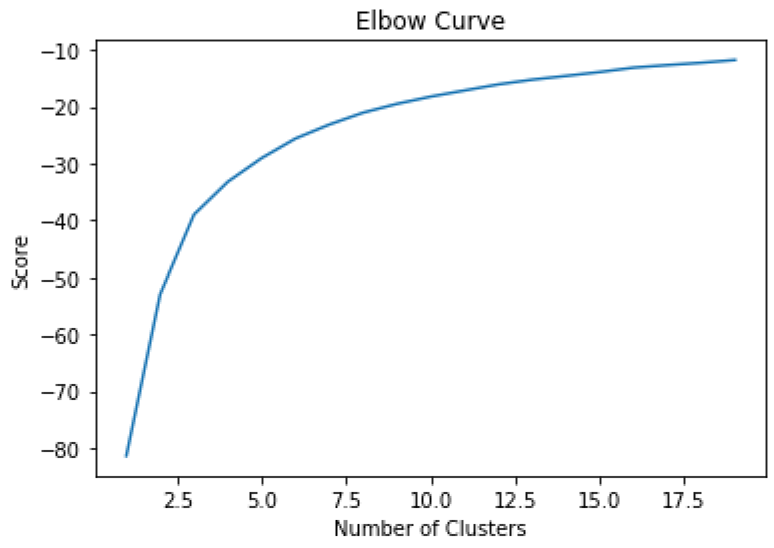
	avg_energy	temperatureMax	dewPoint	cloudCover	windSpeed	pressure	visibility	humidity	uvIndex	moonPhase
avg_energy	1.000000	-0.846965	-0.755901	0.241779	0.149624	-0.028851	-0.246404	0.361237	-0.733171	-0.031716
temperatureMax	-0.846965	1.000000	0.865038	-0.333409	-0.153602	0.118933	0.259108	-0.404899	0.696497	0.003636
dewPoint	-0.755901	0.865038	1.000000	-0.025207	-0.092212	-0.028121	0.042633	0.055514	0.486692	-0.008239

cloudCover	0.241779	-0.333409	-0.025207	1.000000	0.170235	-0.101079	-0.330177	0.480056	-0.248695	-0.062126
windSpeed	0.149624	-0.153602	-0.092212	0.170235	1.000000	-0.344354	0.281088	-0.042391	-0.152634	-0.023273
pressure	-0.028851	0.118933	-0.028121	-0.101079	-0.344354	1.000000	-0.012508	-0.250941	0.100774	0.038462
visibility	-0.246404	0.259108	0.042633	-0.330177	0.281088	-0.012508	1.000000	-0.578130	0.240485	0.062813
humidity	0.361237	-0.404899	0.055514	0.480056	-0.042391	-0.250941	-0.578130	1.000000	-0.533919	-0.013997
uvIndex	-0.733171	0.696497	0.486692	-0.248695	-0.152634	0.100774	0.240485	-0.533919	1.000000	0.012833
moonPhase	-0.031716	0.003636	-0.008239	-0.062126	-0.023273	0.038462	0.062813	-0.013997	0.012833	1.000000

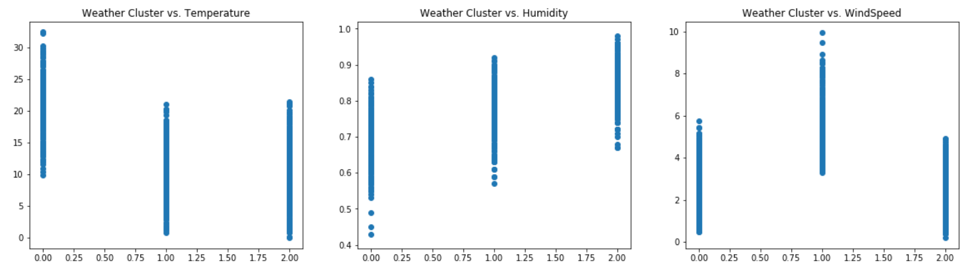
Creating Weather Clusters

The weather information has a lot of variables- which might not all be useful. We will attempt to create weather clusters to see if we can define a weather of the day based on the granular weather data like temperature, precipitation etc.

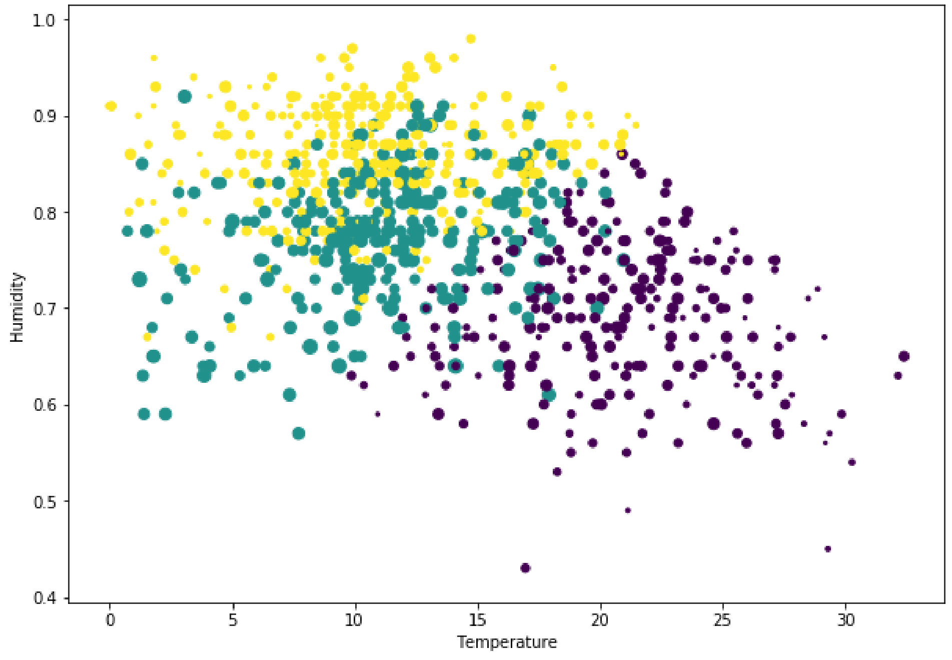
```
#scaling
scaler = MinMaxScaler()
weather_scaled =
scaler.fit_transform(weather_energy[['temperatureMax','humidity','windSpeed']])


# optimum K
Nc = range(1, 20)
kmeans = [KMeans(n_clusters=i) for i in Nc]
kmeans

score = [kmeans[i].fit(weather_scaled).score(weather_scaled) for i in range(len(kmeans))]
score
plt.plot(Nc,score)
plt.xlabel('Number of Clusters')
plt.ylabel('Score')
plt.title('Elbow Curve')
plt.show()
```



```
kmeans = KMeans(n_clusters=3, max_iter=600, algorithm = 'auto')
kmeans.fit(weather_scaled)
weather_energy['weather_cluster'] = kmeans.labels_

# Cluster Relationships with weather variables
plt.figure(figsize=(20,5))
plt.subplot(1, 3, 1)
plt.scatter(weather_energy.weather_cluster,weather_energy.temperatureMax)
plt.title('Weather Cluster vs. Temperature')
plt.subplot(1, 3, 2)
plt.scatter(weather_energy.weather_cluster,weather_energy.humidity)
plt.title('Weather Cluster vs. Humidity')
plt.subplot(1, 3, 3)
plt.scatter(weather_energy.weather_cluster,weather_energy.windSpeed)
plt.title('Weather Cluster vs. WindSpeed')

plt.show()
# put this in a loop
```



```
fig, ax1 = plt.subplots(figsize = (10,7))
ax1.scatter(weather_energy.temperatureMax,
            weather_energy.humidity,
            s = weather_energy.windSpeed*10,
            c = weather_energy.weather_cluster)
ax1.set_xlabel('Temperature')
ax1.set_ylabel('Humidity')
plt.show()
```



UK Bank Holidays

```
holiday = pd.read_csv('../input/uk_bank_holidays.csv')
holiday['Bank holidays'] = pd.to_datetime(holiday['Bank
holidays'],format='%Y-%m-%d').dt.date
holiday.head(4)
```

	Bank holidays	Type	
0	2012-12-26	Boxing Day	
1	2012-12-25	Christmas Day	
2	2012-08-27	Summer bank holiday	
3	2012-05-06	Queen?s Diamond Jubilee (extra bank holiday)	

Creating a holiday indicator on weather data

```
weather_energy = weather_energy.merge(holiday, left_on = 'day',right_on =
'Bank holidays',how = 'left')
weather_energy['holiday_ind'] = np.where(weather_energy['Bank
holidays'].isna(),0,1)
```

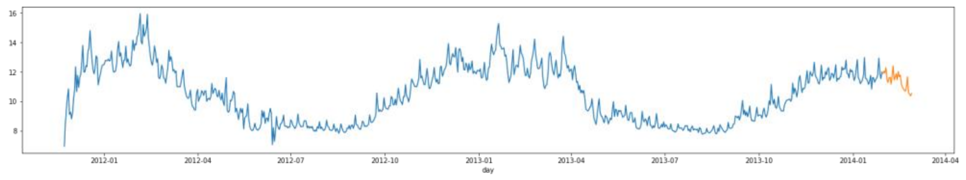
ARIMAX

```
weather_energy['Year'] = pd.DatetimeIndex(weather_energy['day']).year
weather_energy['Month'] = pd.DatetimeIndex(weather_energy['day']).month
weather_energy.set_index(['day'],inplace=True)
```

Subset for required columns and 70-30 train-test split

```
model_data =
weather_energy[['avg_energy','weather_cluster','holiday_ind']]
# train = model_data.iloc[0:round(len(model_data)*0.90)]
# test = model_data.iloc[len(train)-1:]
train = model_data.iloc[0:(len(model_data)-30)]
test = model_data.iloc[len(train):(len(model_data)-1)]
```

```
train['avg_energy'].plot(figsize=(25,4))
test['avg_energy'].plot(figsize=(25,4))
<matplotlib.axes._subplots.AxesSubplot at 0x7f1d002dc9e8>
```

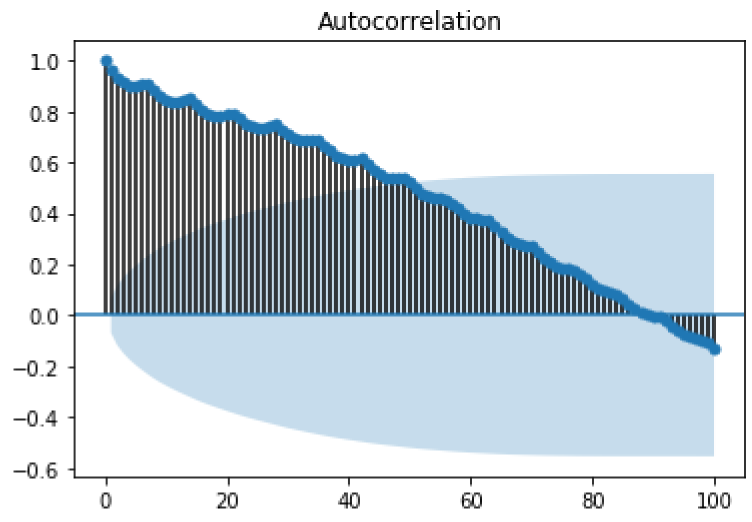


```
test.head(1)
```

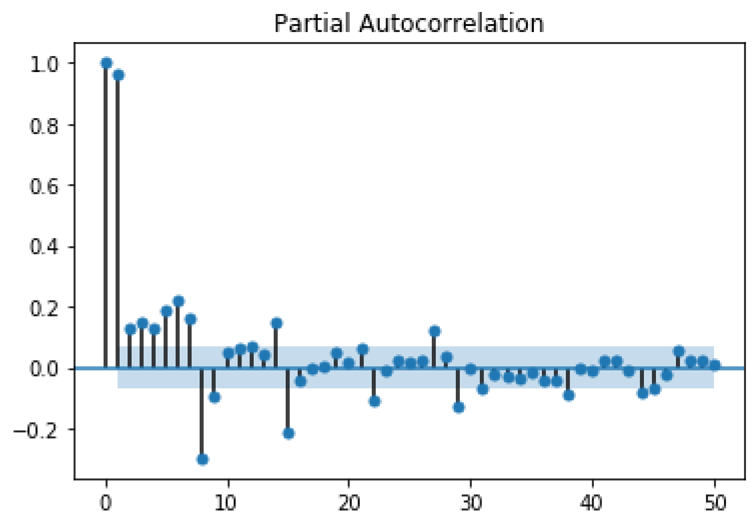
	avg_energy	weather_cluster	holiday_ind
day			
2014-01-30	11.886982	2	0

ACF PACF

```
plot_acf(train.avg_energy,lags=100)
plt.show()
```



```
plot_pacf(train.avg_energy,lags=50)
plt.show()
```



Autocorrelation plot shows gradual decay while Partial AutoCorrelation shows that there is a sharp drop after 1st lag. This means that most of the higher-order autocorrelations are effectively explained by the k = 1 lag. Therefore, the series displays AR 'signature'

Dickey Fuller's Test

p is greater than 0.05 therefore the data is not stationary. After differencing, p < 0.05.

```
t = sm.tsa.adfuller(train.avg_energy, autolag='AIC')
pd.Series(t[0:4], index=['Test Statistic','p-value','#Lags Used','Number
of Observations Used'])
```

```
Test Statistic                   -1.872794
p-value                           0.344966
#Lags Used                       21.000000
Number of Observations Used     776.000000
dtype: float64
```

```
# function for differencing
def difference(dataset, interval):
    diff = list()
    for i in range(interval, len(dataset)):
        value = dataset.iloc[i] - dataset.iloc[i - interval]
        diff.append(value)
    return diff


t  = sm.tsa.adfuller(difference(train.avg_energy,1), autolag='AIC')
pd.Series(t[0:4], index=['Test Statistic','p-value','#Lags Used','Number
of Observations Used'])
```

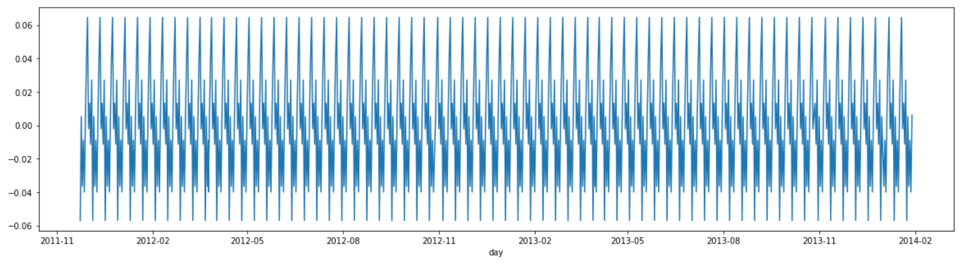
```
Test Statistic                 -6.715004e+00
p-value                         3.600554e-09
#Lags Used                      2.000000e+01
Number of Observations Used     7.760000e+02
dtype: float64
```

Seasonal Decomposition

The seasonal component is quite low while the trend is quite strong with obvious dips in electricity consumption during summers i.e. April to September. This may be attributed to longer days during summer.

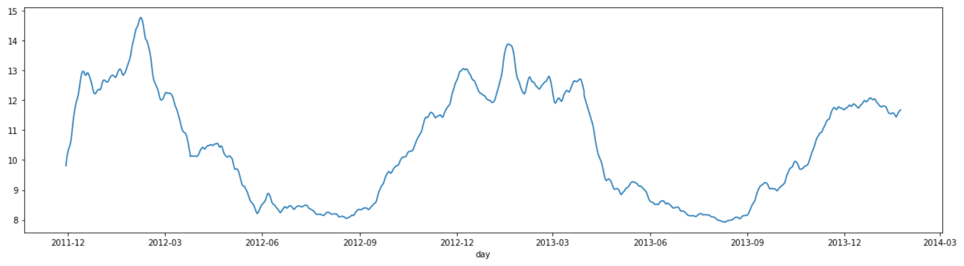
```
s = sm.tsa.seasonal_decompose(train.avg_energy,freq=12)

s.seasonal.plot(figsize=(20,5))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1d0021b400>
```



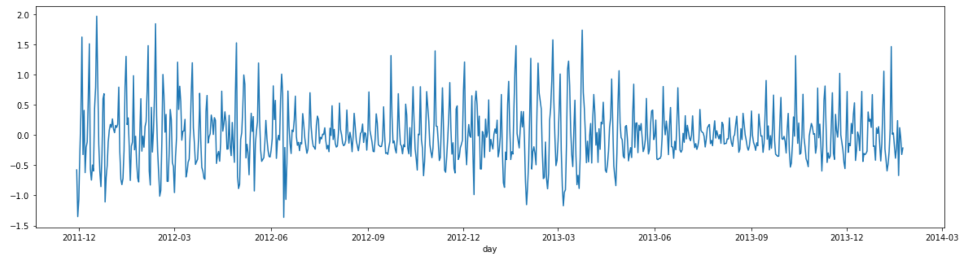
```
s.trend.plot(figsize=(20,5))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1d00108e10>
```



```
s.resid.plot(figsize=(20,5))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1d000f15c0>
```



```
endog = train['avg_energy']
exog = sm.add_constant(train[['weather_cluster','holiday_ind']])

mod = sm.tsa.statespace.SARIMAX(endog=endog, exog=exog,
order=(7,1,1),seasonal_order=(1,1, 0, 12),trend='c')
model_fit = mod.fit()
model_fit.summary()
```

Statespace Model Results

Dep. Variable:	avg_energy	No. Observations:	798
Model:	SARIMAX(7, 1, 1)x(1, 1, 0, 12)	Log Likelihood	-649.420
Date:	Tue, 11 Dec 2018	AIC	1326.841
Time:	10:40:33	BIC	1392.160
Sample:	0	HQIC	1351.956
	- 798		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	-0.0064	0.017	-0.379	0.705	-0.039	0.027
const	-3.162e-08	2.89e-10	-109.267	0.000	-3.22e-08	-3.1e-08
weather_cluster	0.0031	0.023	0.134	0.893	-0.042	0.048
holiday_ind	-0.0344	0.088	-0.392	0.695	-0.207	0.138
ar.L1	-0.0011	0.086	-0.013	0.990	-0.170	0.168
ar.L2	-0.1545	0.032	-4.841	0.000	-0.217	-0.092

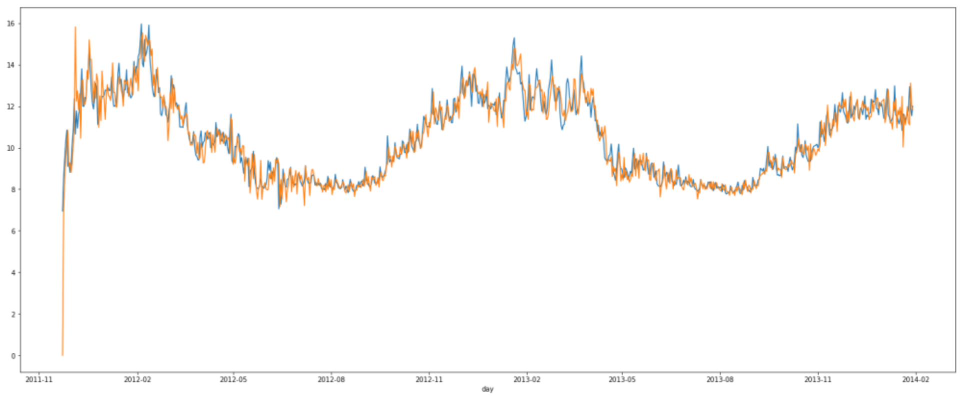
ar.L3	-0.1434	0.038	-3.763	0.000	-0.218	-0.069
ar.L4	-0.1513	0.038	-3.987	0.000	-0.226	-0.077
ar.L5	-0.1632	0.040	-4.107	0.000	-0.241	-0.085
ar.L6	0.0087	0.036	0.239	0.811	-0.062	0.080
ar.L7	0.3526	0.029	12.333	0.000	0.297	0.409
ma.L1	-0.1860	0.091	-2.043	0.041	-0.365	-0.008
ar.S.L12	-0.4836	0.032	-14.939	0.000	-0.547	-0.420
sigma2	0.3041	0.013	24.110	0.000	0.279	0.329

Ljung-Box (Q):	221.13	Jarque-Bera (JB):	45.29
Prob(Q):	0.00	Prob(JB):	0.00
Heteroskedasticity (H):	0.53	Skew:	-0.16
Prob(H) (two-sided):	0.00	Kurtosis:	4.13

```
Warnings:
[1] Covariance matrix calculated using the outer product of gradients
(complex-step).
[2] Covariance matrix is singular or near-singular, with condition number
5.86e+16. Standard errors may be unstable.
```

Model Fit

```
train['avg_energy'].plot(figsize=(25,10))
model_fit.fittedvalues.plot()
plt.show()
```



Prediction

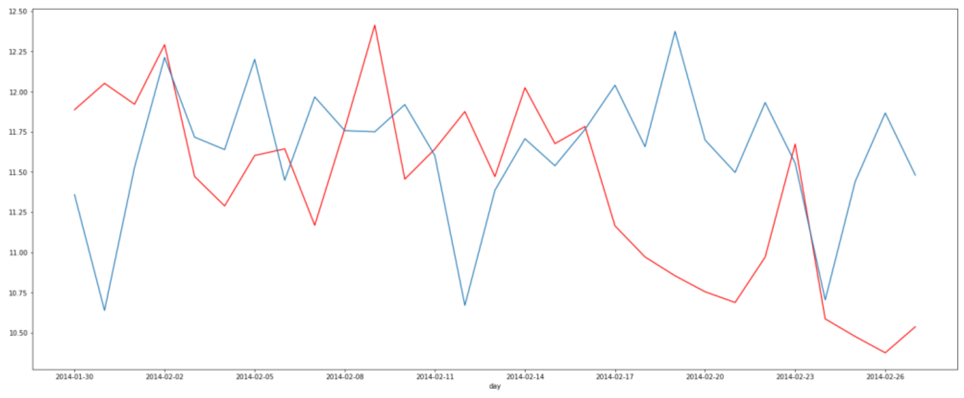
```
predict = model_fit.predict(start = len(train),end = len(train)+len(test)-
1,exog = sm.add_constant(test[['weather_cluster','holiday_ind']]))
test['predicted'] = predict.values
test.tail(5)
```

	avg_energy	weather_cluster	holiday_ind	predicted
day				
2014-02-23	11.673756	1	0	11.554959
2014-02-24	10.586235	1	0	10.704375
2014-02-25	10.476498	1	0	11.441180
2014-02-26	10.375366	1	0	11.866796
2014-02-27	10.537250	1	0	11.480418

```
test['residual'] = abs(test['avg_energy']-test['predicted'])
MAE = test['residual'].sum()/len(test)
MAPE = (abs(test['residual'])/test['avg_energy']).sum()*100/len(test)
print("MAE:", MAE)
print("MAPE:", MAPE)
```

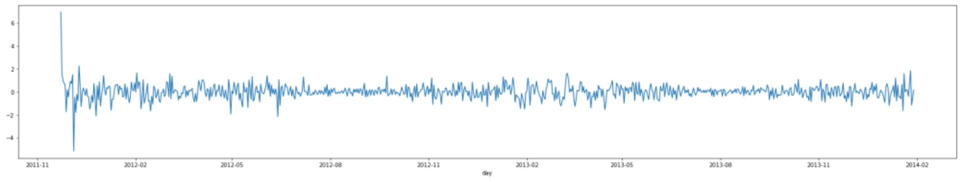
```
MAE: 0.5853190227226445
MAPE: 5.237822685938685
```

```
test['avg_energy'].plot(figsize=(25,10),color = 'red')
test['predicted'].plot()
plt.show()
```

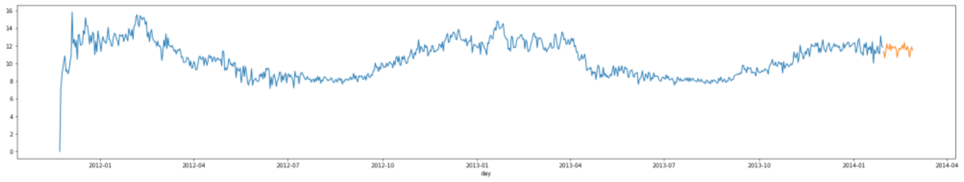


```
model_fit.resid.plot(figsize= (30,5))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1cf7dc54e0>
```



```
model_fit.fittedvalues.plot(figsize = (30,5))
test.predicted.plot()

<matplotlib.axes._subplots.AxesSubplot at 0x7f1d0021bba8>
```



```
test['predicted'].tail(5)

day
2014-02-23    11.554959
2014-02-24    10.704375
2014-02-25    11.441180
2014-02-26    11.866796
2014-02-27    11.480418
Name: predicted, dtype: float64
```

LSTM

Using lags of upto 7 days we are going to convert this into a supervised problem. I have taken the function to create lags from this tutorial by Jason Brownlee. He has also applied the same to convert multivariate data to a supervised dataframe which he has in turn applied LSTM on.

```
np.random.seed(11)
dataframe = weather_energy.loc[:,'avg_energy']
dataset = dataframe.values
dataset = dataset.astype('float32')

# convert series to supervised learning
def series_to_supervised(data, n_in=1, n_out=1, dropnan=True):
    n_vars = 1
    df = pd.DataFrame(data)
    cols, names = list(), list()
    # input sequence (t-n, ... t-1)
    for i in range(n_in, 0, -1):
        cols.append(df.shift(i))
        names += [('var%d(t-%d)' % (j+1, i)) for j in range(n_vars)]
    # forecast sequence (t, t+1, ... t+n)
    for i in range(0, n_out):
        cols.append(df.shift(-i))
        if i == 0:
            names += [('var%d(t)' % (j+1)) for j in range(n_vars)]
        else:
            names += [('var%d(t+%d)' % (j+1, i)) for j in range(n_vars)]
    # put it all together
    agg = pd.concat(cols, axis=1)
    agg.columns = names
    # drop rows with NaN values
    if dropnan:
        agg.dropna(inplace=True)
    return agg

reframed = series_to_supervised(dataset, 7,1)
reframed.head(3)
```

	var1(t-7)	var1(t-6)	var1(t-5)	var1(t-4)	var1(t-3)	var1(t-2)	var1(t-1)	var1(t)
7	6.952693	8.536480	9.499782	10.267707	10.850805	9.103382	9.274873	8.813513
8	8.536480	9.499782	10.267707	10.850805	9.103382	9.274873	8.813513	9.227707
9	9.499782	10.267707	10.850805	9.103382	9.274873	8.813513	9.227707	10.145910

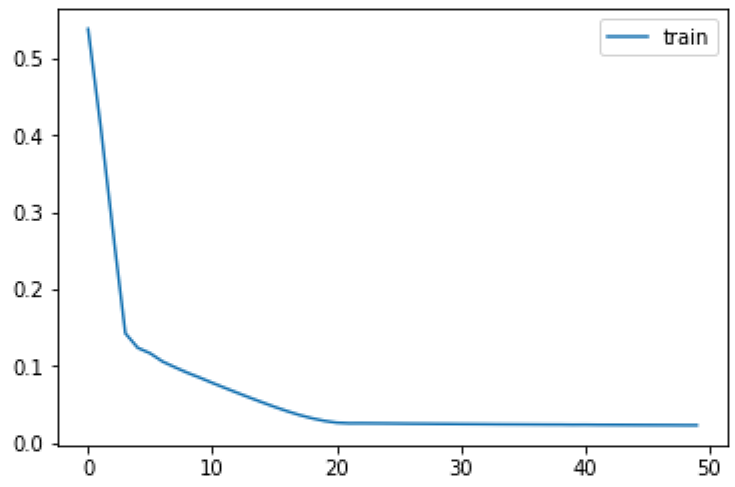
```
reframed['weather_cluster'] = weather_energy.weather_cluster.values[7:]
reframed['holiday_ind']= weather_energy.holiday_ind.values[7:]


reframed = reframed.reindex(['weather_cluster', 'holiday_ind','var1(t-7)',
'var1(t-6)', 'var1(t-5)', 'var1(t-4)', 'var1(t-3)','var1(t-2)', 'var1(t-
1)', 'var1(t)'], axis=1)
reframed = reframed.values
```

Normalization

```
scaler = MinMaxScaler(feature_range=(0, 1))
reframed = scaler.fit_transform(reframed)
```

```
# split into train and test sets
train = reframed[:(len(reframed)-30), :]
test = reframed[(len(reframed)-30):len(reframed), :]

train_X, train_y = train[:, :-1], train[:, -1]
test_X, test_y = test[:, :-1], test[:, -1]

# reshape input to be 3D [samples, timesteps, features]
train_X = train_X.reshape((train_X.shape[0], 1, train_X.shape[1]))
test_X = test_X.reshape((test_X.shape[0], 1, test_X.shape[1]))
print(train_X.shape, train_y.shape, test_X.shape, test_y.shape)
```

```
(791, 1, 9) (791,) (30, 1, 9) (30,)
```



Prediction

```
# make a prediction
yhat = model.predict(test_X)

test_X = test_X.reshape(test_X.shape[0], test_X.shape[2])


# invert scaling for forecast
inv_yhat = np.concatenate((yhat, test_X), axis=1)
inv_yhat = scaler.inverse_transform(inv_yhat)
```

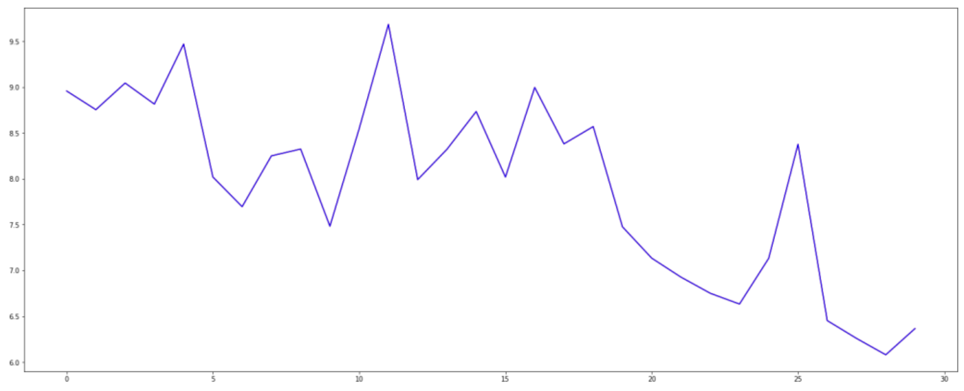
```
# invert scaling for actual
test_y = test_y.reshape((len(test_y), 1))
inv_y = np.concatenate((test_y, test_X), axis=1)
inv_y = scaler.inverse_transform(inv_y)
```

Performance

```
act = [i[9] for i in inv_y] # last element is the predicted average energy
pred = [i[9] for i in inv_yhat] # last element is the actual average energy

# calculate RMSE
import math
rmse = math.sqrt(mean_squared_error(act, pred))
print('Test RMSE: %.3f' % rmse)
```

```
Test RMSE: 0.000
```

```
predicted_lstm = pd.DataFrame({'predicted':pred,'avg_energy':act})
predicted_lstm['avg_energy'].plot(figsize=(25,10),color = 'red')
predicted_lstm['predicted'].plot(color = 'blue')
plt.show()
```



Conclusion:

- In conclusion, the role of the energy consumer is becoming increasingly significant in our rapidly evolving world. As we face challenges such as climate change, resource depletion, and the need for sustainable energy sources, consumers hold the key to driving positive change. Through informed choices, energy efficiency measures, and a growing emphasis on renewable energy sources, consumers can contribute to reducing their carbon footprint and promoting a more sustainable future.

- Moreover, the advent of smart technology and the Internet of Things has empowered consumers with real-time information and control over their energy consumption. This newfound awareness and control not only benefit individuals and households but also have a collective impact on global energy trends.

- It is essential for consumers to stay informed about energy-efficient practices, take advantage of government incentives, and support policies that promote clean and renewable energy sources. By doing so, they not only reduce their energy costs but also play a crucial role in the transition to a cleaner and more sustainable energy future.

- In essence, energy consumers are not just passive users but active agents of change in the energy landscape. Their choices and behaviors can lead us towards a more sustainable and environmentally responsible energy future, benefiting not only the planet but also future generations.