NILMTK Rapid Experimentation API

Importando bibliotecas

In [1]:

```
from matplotlib import rcParams
import matplotlib.pyplot as plt
import pandas as pd
import nilmtk
from nilmtk import MeterGroup
from nilmtk.api import API
import warnings
warnings.filterwarnings("ignore")

plt.style.use('ggplot')
rcParams['figure.figsize'] = (13, 10)

import pathlib
pathlib.Path().resolve()
```

Out[1]:

```
PosixPath('/home/hb/projetos/nilmtk')
```

Convertendo a base de dados

In [20]:

```
from nilmtk.dataset_converters import convert_redd
convert_redd('./BD/REDD/low_freq/', './data/redd_c12.h5')
```

```
Loading house 1... 1 2 11
Loading house 2... 1 2 3 4 5 6 7 8 9 10 11
Loading house 3... 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
Loading house 4... 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Loading house 5... 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
Loading house 6... 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
Loaded metadata
Done converting YAML metadata to HDF5!
Done converting REDD to HDF5!
```

Importando a base de dados

In [21]:

```
from nilmtk import DataSet
from nilmtk.utils import print_dict

redd = DataSet('./data/redd_c12.h5')
#iawe = DataSet('/data/iawe.h5')

print_dict(redd.metadata)
print_dict(redd.buildings)
```

- **name**: REDD
- **long_name**: The Reference Energy Disaggregation Data set
- **creators**:
  - Kolter, Zico
  - Johnson, Matthew
- **publication_date**: 2011
- **institution**: Massachusetts Institute of Technology (MIT)
- **contact**: zkolter@cs.cmu.edu
- **description**: Several weeks of power data for 6 different homes.
- **subject**: Disaggregated power demand from domestic buildings.
- **number_of_buildings**: 6
- **timezone**: US/Eastern
- **geo_location**:
  - **locality**: Massachusetts
  - **country**: US
  - **latitude**: 42.360091
  - **longitude**: -71.09416
- **related_documents**:
  - http://redd.csail.mit.edu (http://redd.csail.mit.edu)
  - J. Zico Kolter and Matthew J. Johnson. REDD: A public data set for energy disaggregation research. In proceedings of the SustKDD workshop on Data Mining Applications in Sustainability, 2011. http://redd.csail.mit.edu/kolter-kddsust11.pdf (http://redd.csail.mit.edu/kolter-kddsust11.pdf)
- **schema**: https://github.com/nilmtk/nilm_metadata/tree/v0.2 (https://github.com/nilmtk/nilm_metadata/tree/v0.2)
- **meter_devices**:
  - **eMonitor**:
    - **model**: eMonitor
    - **manufacturer**: Powerhouse Dynamics
    - **manufacturer_url**: http://powerhousedynamics.com (http://powerhousedynamics.com)
    - **description**: Measures circuit-level power demand. Comes with 24 CTs. This FAQ page suggests the eMonitor measures real (active) power: http://www.energycircle.com/node/14103 (http://www.energycircle.com/node/14103) although the REDD readme.txt says all channels record apparent power.
    - **sample_period**: 3
    - **max_sample_period**: 50
    - **measurements**:
      - {'physical_quantity': 'power', 'type': 'active', 'upper_limit': 5000, 'lower_limit': 0}
    - **wireless**: False
  - **REDD_whole_house**:
    - **description**: REDD's DIY power meter used to measure whole-home AC waveforms at high frequency. To quote from their paper: "CTs from TED (http://www.theenergydetective.com (http://www.theenergydetective.com)) to measure current in the power mains, a Pico TA041 oscilloscope probe (http://www.picotechnologies.com (http://www.picotechnologies.com)) to measure voltage for one of the two phases in the home, and a National Instruments NI-9239 analog to digital converter to transform both these analog signals to digital readings. This A/D converter has 24 bit resolution with noise of approximately 70 µV, which determines the noise level of our current and voltage readings: the TED CTs are rated for 200 amp circuits and a maximum of 3 volts, so we are able to differentiate between currents of approximately ((200))(70 × 10−6)/(3) = 4.66mA, corresponding to power changes of about 0.5 watts. Similarly, since we use a 1:100 voltage stepdown in the oscilloscope probe, we can detect voltage differences of about 7mV."
    - **sample_period**: 1
    - **max_sample_period**: 30
    - **measurements**:
      - {'physical_quantity': 'power', 'type': 'apparent', 'upper_limit': 50000, 'lower_limit': 0}
    - **wireless**: False

- **1**: Building(instance=1, dataset='REDD')
- **2**: Building(instance=2, dataset='REDD')
- **3**: Building(instance=3, dataset='REDD')
- **4**: Building(instance=4, dataset='REDD')
- **5**: Building(instance=5, dataset='REDD')
- **6**: Building(instance=6, dataset='REDD')

Carregando exemplo de uma casa/eletrodoméstico

In [23]:

```
build = 1
elec = redd.buildings[build].elec
elec.mains().power_series_all_data().head()

#sns.set_palette("Set3", n_colors=12)
# Set a threshold to remove residual power noise when devices are off
#elec.plot_when_on(on_power_threshold = 40) # Plot appliances when they are in use¶

```

```
Loading data for meter ElecMeterID(instance=2, building=1, dataset='REDD')
Done loading data all meters for this chunk.
```

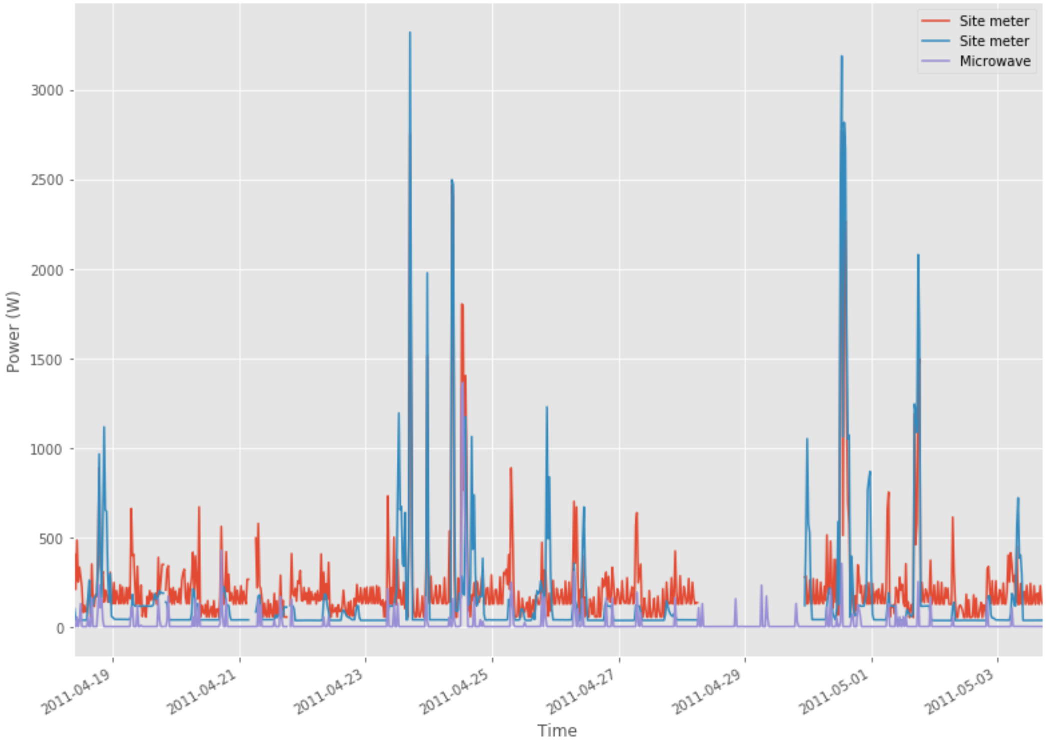
Out[23]:

```
2011-04-18 09:22:09-04:00    342.820007
2011-04-18 09:22:10-04:00    344.559998
2011-04-18 09:22:11-04:00    345.140015
2011-04-18 09:22:12-04:00    341.679993
2011-04-18 09:22:13-04:00    341.029999
Freq: S, Name: (power, apparent), dtype: float32
```

In [31]:

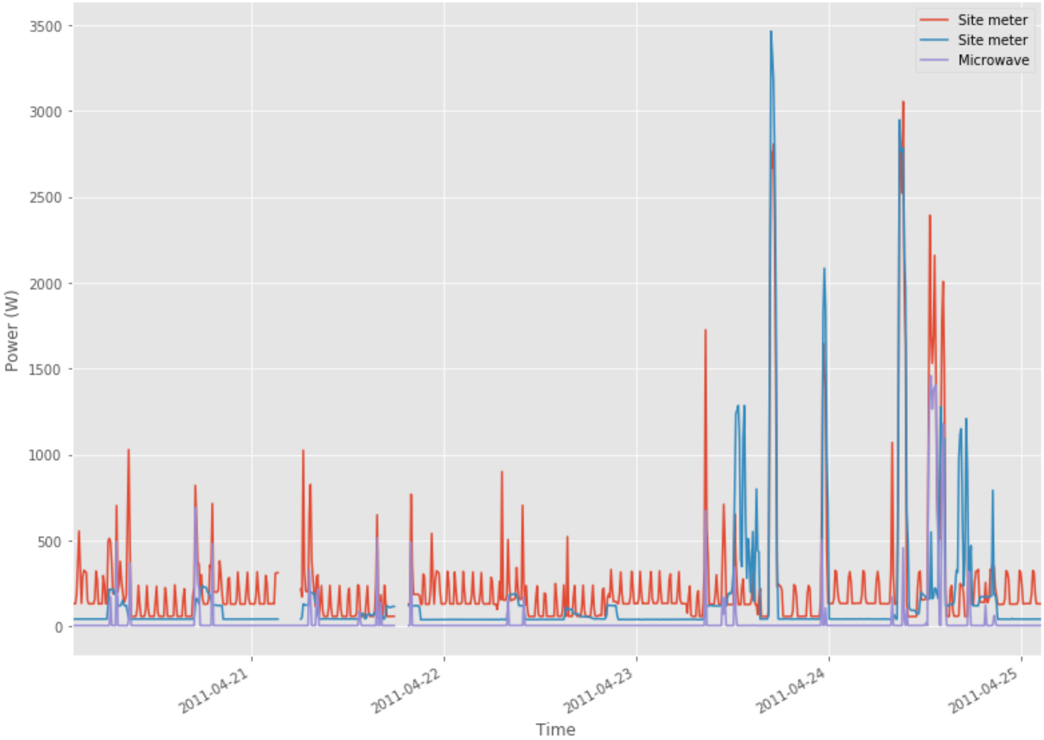
```
microwave = elec['microwave']
#fridge.available_columns()
print(next(microwave.load()).head())

from nilmtk.elecmeter import ElecMeterID

meter1 = elec[ElecMeterID(instance=0, building=build, dataset='REDD')]

redd.set_window(start='2011-04-16 05:11:27', end='2011-05-04 00:19:54')
meter1.plot()
elec['microwave'].plot()
plt.xlabel("Time");
```

```
physical_quantity          power
type                      active
2011-04-20 01:40:01-04:00    4.0
2011-04-20 01:40:04-04:00    4.0
2011-04-20 01:40:08-04:00    4.0
2011-04-20 01:40:11-04:00    4.0
2011-04-20 01:40:20-04:00    4.0
...                          ...
2011-04-25 02:39:44-04:00    4.0
2011-04-25 02:39:47-04:00    4.0
2011-04-25 02:39:50-04:00    4.0
2011-04-25 02:39:54-04:00    4.0
2011-04-25 02:39:58-04:00    4.0

[110977 rows x 1 columns]
```



In [25]:

```
redd.set_window(start='2011-04-20 01:40:00', end='2011-04-25 02:40:00')
meter1.plot() # 1 segundo
elec['microwave'].plot() # 3 segundos
plt.xlabel("Time");
```



In [26]:

```
next(elec.load())
```

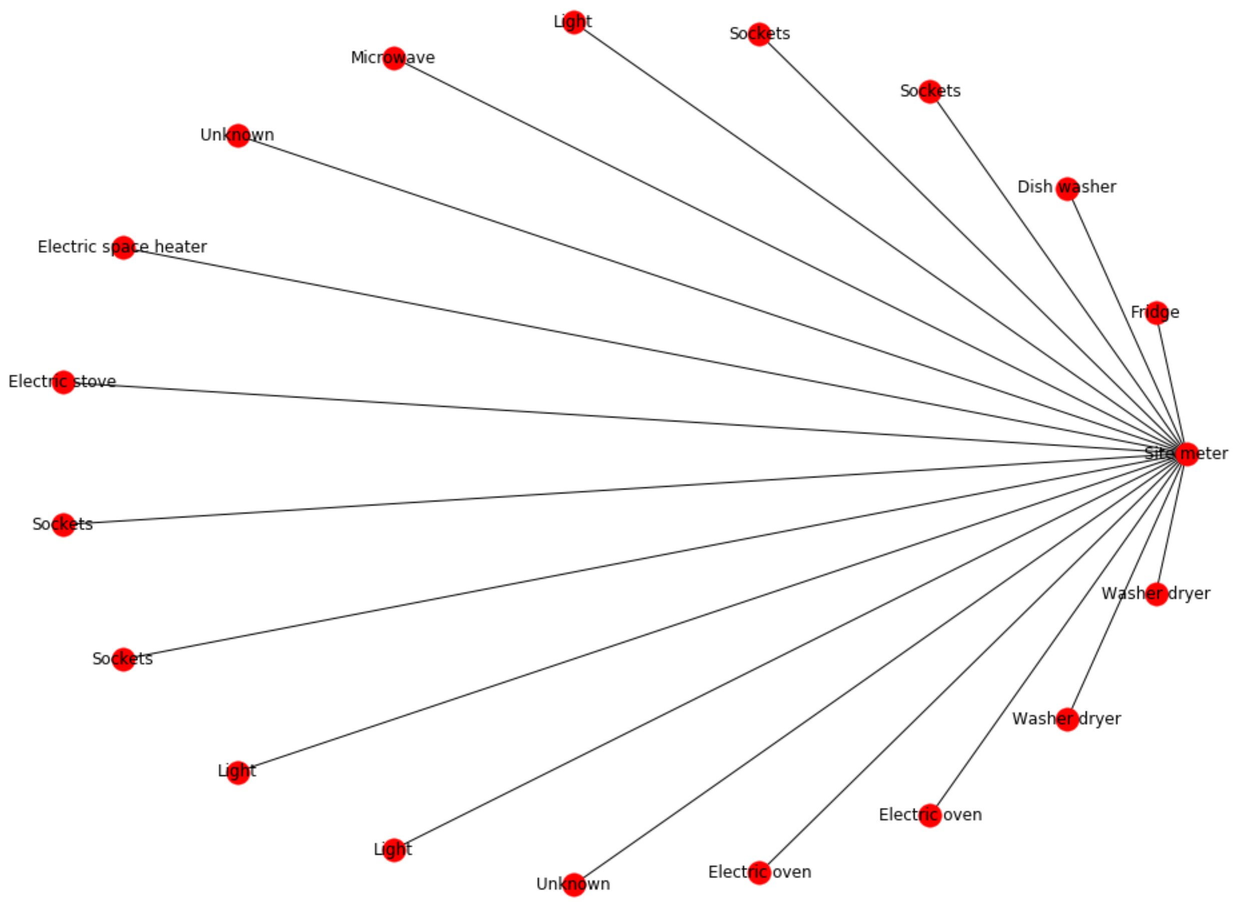
```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-26-d3dd7fad46f3> in <module>
----> 1 next(elec.load())

~/anaconda3/envs/nilmtk-env/lib/python3.7/site-packages/nilmtk/metergroup.py in load(self, **kwargs)
    697         # Handle kwargs
    698         sample_period = kwargs.setdefault('sample_period', self.sample_period())
--> 699         sections = kwargs.pop('sections', [self.get_timeframe()])
    700         chunksize = kwargs.pop('chunksize', MAX_MEM_ALLOWANCE_IN_BYTES)
    701         duration_threshold = sample_period * chunksize

~/anaconda3/envs/nilmtk-env/lib/python3.7/site-packages/nilmtk/metergroup.py in get_timeframe(self)
   1386             if timeframe is None:
   1387                 timeframe = meter.get_timeframe()
-> 1388             elif meter.get_timeframe().empty:
   1389                 pass
   1390             else:

~/anaconda3/envs/nilmtk-env/lib/python3.7/site-packages/nilmtk/elecmeter.py in get_timeframe(self)
     92     def get_timeframe(self):
     93         self._check_store()
---> 94         return self.store.get_timeframe(key=self.key)
     95 
     96     def _check_store(self):

~/anaconda3/envs/nilmtk-env/lib/python3.7/site-packages/nilmtk/docinherit.py in f(*args, **kwargs)
     51         def f(*args, **kwargs):
     52             if obj:
---> 53                 return self.mthd(obj, *args, **kwargs)
     54             else:
     55                 return self.mthd(*args, **kwargs)

~/anaconda3/envs/nilmtk-env/lib/python3.7/site-packages/nilmtk/datastore/hdfdatastore.py in get_timeframe(self,
 key)
    220         nilmtk.TimeFrame of entire table after intersecting with self.window.
    221         """
--> 222         data_start_date = self.store.select(key, [0]).index[0]
    223         data_end_date = self.store.select(key, start=-1).index[0]
    224         timeframe = TimeFrame(data_start_date, data_end_date)

~/anaconda3/envs/nilmtk-env/lib/python3.7/site-packages/pandas/io/pytables.py in select(self, key, where, start,
stop, columns, iterator, chunksize, auto_close, **kwargs)
    755         group = self.get_node(key)
    756         if group is None:
--> 757             raise KeyError("No object named {key} in the file".format(key=key))
    758 
    759         # create the storer and axes

KeyError: 'No object named /building1/elec/meter5 in the file'
```

In [27]:

```
elec.draw_wiring_graph()
```

Out[27]:

```
(<networkx.classes.digraph.DiGraph at 0x7f5c8a393b90>,
 <matplotlib.axes._axes.Axes at 0x7f5ce8471f50>)
```



Importamos os algoritmos que desejamos executar os experimentos:

- Mean: Mean Algorithm
- Hart's Algorithm
- CO: Combinatorial Optimization
- Discriminative Sparse Coding
- Additive Factorial Hidden Markov Model
- Additive Factorial Hidden Markov Model with Signal Aggregate Constraints
- DSC: Discriminative Sparse Coding
- RNN: Long short-term memory - LSTM
- DAE: Denoising Auto Encoder
- Seq2Point*
- Seq2Seq
- WindowGRU/Online GRU: Similar a LSTM, mas usa Gated Recurrent Unit (GRU)
- ELM

In [28]:

```
from nilmtk.disaggregate import Mean,CO,Hart85
# from nilmtk_contrib.disaggregate import AFHMM,AFHMM_SAC,DSC,RNN,Seq2Point,Seq2Seq,DAE,WindowGRU
from nilmtk_contrib.disaggregate import RNN


```

Em seguida, inserimos os valores para os diferentes parâmetros no dicionário. Como precisamos de vários aparelhos, inserimos os nomes de todos os aparelhos necessários no parâmetro *'appliances'*.

Métricas: [https://github.com/nilmtk/nilmtk/blob/master/nilmtk/losses.py (https://github.com/nilmtk/nilmtk/blob/master/nilmtk/losses.py)](https://github.com/nilmtk/nilmtk/blob/master/nilmtk/losses.py)

In [29]:

```
experiment1 = {
  'power': {'mains': ['apparent'],'appliance': ['active']},
  'sample_rate': 1,
  'appliances': ['microwave'],
  'methods': {
      'Mean':Mean({}),
      "CO":CO({}),
      'Hart85':Hart85({}),
      'RNN':RNN({'n_epochs':1,'batch_size':128})
      #"AFHMM":AFHMM({}),
      #"AFHMM_SAC":AFHMM_SAC({}),
  },
  'train': {
    'datasets': {
        'Redd': {
            'path': './data/redd_c12.h5',
            'buildings': {
                1: {
                    'start_time': '2011-04-19',
                    'end_time': '2011-05-09'
                    }
                }
            }
        }
    },
  'test': {
    'datasets': {
        'Redd': {
            'path': './data/redd_c12.h5',
            'buildings': {
                1: {
                    'start_time': '2011-05-10',
                    'end_time': '2011-05-23'
                    }
                }
            }
        },
        'metrics':['rmse', 'mae', 'relative_error', 'r2score', 'nde', 'nep', 'f1score']
    }
}
```

In this example experimental setup, we have set the *sample rate* at 60Hz and use Combinatorial Optimisation to disaggregate the required appliances from building 10 in the dataport dataset with the *RMSE* metric to measure the accuracy. We also specify the dates for training and testing

Next we provide this experiment dictionary as input to the API.

In [30]:

```
api_results_experiment_1 = API(experiment1)
```

```
Joint Testing for all algorithms
Loading data for  Redd  dataset
Loading data for meter ElecMeterID(instance=2, building=1, dataset='REDD')
Done loading data all meters for this chunk.
Dropping missing values
Generating predictions for : Mean
Generating predictions for : CO
...............CO disaggregate_chunk running............
Generating predictions for : Hart85ave'
Finding Edges, please wait ...
Edge detection complete.
Creating transition frame ...
Transition frame created.
Creating states frame ...
States frame created.
Finished.
Generating predictions for : RNN
............  rmse  .............
                Mean         CO      Hart85        RNN
microwave  125.13442  406.63476  168.727064  87.431072
............  mae  .............
               Mean          CO     Hart85        RNN
microwave  28.47261  172.789413  80.320747  10.684411
............  relative_error  .............
               Mean        CO    Hart85       RNN
microwave  1.214042  0.933382  6.082606  0.251585
............  r2score  .............
               Mean        CO   Hart85       RNN
microwave -0.004053 -9.602589 -0.82546  0.509844
............  nde  .............
               Mean        CO    Hart85       RNN
microwave  0.995335  3.234423  1.342076  0.695438
............  nep  .............
               Mean         CO   Hart85       RNN
microwave  1.963248  11.914206  5.53829  0.736713
............  f1score  .............
               Mean        CO    Hart85       RNN
microwave  0.022326  0.022326  0.058907  0.420854
```

raiz do erro quadrático médio (RMSE) e o erro médio absoluto (MAE)

Quanto menor o seu valor, melhor é o modelo, já que a previsão se mostra mais próxima ao valor real. Comparando as duas métricas têm se que o RMSE penaliza desvios grandes, enquanto o MAE tem pesos iguais para todos os desvios.

We can observe the prediction vs. truth graphs in the above cell. The accuracy metrics can be accessed using the following commands:

In [8]:

```
errors_keys = api_results_experiment_1.errors_keys
errors = api_results_experiment_1.errors
for i in range(len(errors)):
    print (errors_keys[i])
    print (errors[i])
    print ("\n\n")
```

```
Redd_1_rmse
                Mean          CO      Hart85        RNN
microwave  125.13442  406.084447  168.727064  91.430782



Redd_1_mae
               Mean          CO     Hart85        RNN
microwave  28.47261  167.383987  80.320747  19.216148



Redd_1_relative_error
               Mean        CO    Hart85      RNN
microwave  1.214042  0.928344  6.082606  0.55415



Redd_1_r2score
               Mean        CO   Hart85       RNN
microwave -0.004053 -9.573909 -0.82546  0.463972



Redd_1_nde
               Mean        CO    Hart85       RNN
microwave  0.995335  3.230046  1.342076  0.727252



Redd_1_nep
               Mean         CO   Hart85       RNN
microwave  1.963248  11.541489  5.53829  1.324995



Redd_1_f1score
               Mean        CO    Hart85       RNN
microwave  0.022326  0.022326  0.058907  0.225944
```

In [ ]:

```

```