Group E Assignment 5: NILMTK

Exercise 1

Load UKDALE data into memory and print out the metadata

In [1]:
```
from nilmtk import DataSet
from nilmtk.utils import print_dict
from nilmtk.timeframe import TimeFrame
import pandas as pd

ukdale = DataSet('./data/ukdale.h5')
```

1.2 Print out Metadata

In [2]:
```
print_dict(ukdale.metadata)
```

- **description_of_subjects**: 4 MSc students and 1 PhD student.
- **meter_devices**:
  - **EcoManagerTxPlug**:
    - **max_sample_period**: 120
    - **model_url**: https://shop.edfenergy.com/Item.aspx?id=540 (https://shop.edfenergy.com/Item.aspx?id=540)
    - **wireless_configuration**:
      - **base**: creators: [Jack Kelly] model: rfm_edf_ecomanager model_url: https://github.com/JackKelly/rfm_edf_ecomanager/ (creators: [Jack Kelly] model: rfm_edf_ecomanager model_url: https://github.com/JackKelly/rfm_edf_ecomanager/)
      - **protocol**: custom
      - **carrier_frequency**: 434
    - **brand**: EcoManager
    - **measurements**:
      - {'lower_limit': 0, 'upper_limit': 3300, 'physical_quantity': 'power', 'type': 'active'}
    - **data_logger**:
      - **model**: rfm_ecomanager_logger
      - **creators**:
        - Jack Kelly
      - **model_url**:

https://github.com/JackKelly/rfm_ecomanager_logger (https://github.com/JackKelly/rfm_ecomanager_logger)
    - **sample_period**: 6
    - **wireless**: True
    - **brand_url**: http://www.edfenergy.com/products-services/for-your-home/ecomanager (http://www.edfenergy.com/products-services/for-your-home/ecomanager)
    - **model**: EcoManagerTxPlug
    - **seller**: EDF Energy
    - **manufacturer**: Current Cost / Sailwider
- **CurrentCostTx**:
    - **max_sample_period**: 120
    - **model_url**: http://www.currentcost.com/product-transmitter.html (http://www.currentcost.com/product-transmitter.html)
    - **wireless_configuration**:
        - **base**: creators: [Jack Kelly] model: rfm_edf_ecomanager model_url: https://github.com/JackKelly/rfm_edf_ecomanager/ (creators: [Jack Kelly] model: rfm_edf_ecomanager model_url: https://github.com/JackKelly/rfm_edf_ecomanager/)
        - **protocol**: custom
        - **carrier_frequency**: 434
    - **measurements**:
        - {'lower_limit': 0, 'upper_limit': 25000, 'physical_quantity': 'power', 'type': 'apparent'}
    - **data_logger**:
        - **model**: rfm_ecomanager_logger
        - **creators**:
            - Jack Kelly
        - **model_url**: https://github.com/JackKelly/rfm_ecomanager_logger (https://github.com/JackKelly/rfm_ecomanager_logger)
    - **sample_period**: 6
    - **wireless**: True
    - **model**: CurrentCost Tx
    - **manufacturer**: Current Cost
- **SoundCardPowerMeter**:
    - **max_sample_period**: 3
    - **wireless**: False
    - **model_url**: https://github.com/JackKelly/snd_card_power_meter (https://github.com/JackKelly/snd_card_power_meter)
    - **model**: Sound Card Power Meter
    - **measurements**:
        - {'lower_limit': 0, 'upper_limit': 25000, 'physical_quantity': 'power', 'type': 'active'}

        - {'lower_limit': 0, 'upper_limit': 25000, 'physical_quantity': 'power', 'type': 'apparent'}
        - {'lower_limit': 180, 'upper_limit': 275, 'physical_quantity': 'voltage', 'description': 'RMS voltage'}
      - **manufacturer**: Jack Kelly / Imperial College London
      - **sample_period**: 1
    - **EcoManagerWholeHouseTx**:
      - **max_sample_period**: 120
      - **model_url**: https://shop.edfenergy.com/Item.aspx?id=547 (https://shop.edfenergy.com/Item.aspx?id=547)
      - **wireless_configuration**:
        - **base**: creators: [Jack Kelly] model: rfm_edf_ecomanager model_url: https://github.com/JackKelly/rfm_edf_ecomanager/ (creators: [Jack Kelly] model: rfm_edf_ecomanager model_url: https://github.com/JackKelly/rfm_edf_ecomanager/)
        - **protocol**: custom
        - **carrier_frequency**: 434
      - **brand**: EcoManager
      - **measurements**:
        - {'lower_limit': 0, 'upper_limit': 25000, 'physical_quantity': 'power', 'type': 'apparent'}
      - **data_logger**:
        - **model**: rfm_ecomanager_logger
        - **creators**:
          - Jack Kelly
        - **model_url**: https://github.com/JackKelly/rfm_ecomanager_logger (https://github.com/JackKelly/rfm_ecomanager_logger)
      - **sample_period**: 6
      - **wireless**: True
      - **brand_url**: http://www.edfenergy.com/products-services/for-your-home/ecomanager (http://www.edfenergy.com/products-services/for-your-home/ecomanager)
      - **model**: EcoManagerWholeHouseTx
      - **seller**: EDF Energy
      - **site_meter**: True
      - **manufacturer**: Current Cost / Sailwider
- **description**: Appliance-by-appliance and whole-home power demand for 5 UK homes. Appliance power demand was recorded once every 6 seconds. Whole-home power demand was recorded once every 6 seconds for all homes and additionally at 16kHz for homes 1, 2 and 5. Detailed metadata is included.
- **rights_list**:
  - {'name': 'Creative Commons Attribution 4.0 International (CC BY 4.0)', 'uri': 'http://creativecommons.org/licenses/by/4.0/'}
- **long_name**: UK Domestic Appliance-Level Electricity

- **geo_location**:
  - **latitude**: 51.464462
  - **country**: GB
  - **longitude**: -0.076544
  - **locality**: London
- **date**: 2015-01-05
- **timezone**: Europe/London
- **institution**: Imperial College London
- **subject**: Disaggregated domestic electricity demand
- **publisher**: UK Energy Research Centre Energy Data Centre (UKERC EDC)
- **funding**:
  - Jack Kelly's PhD is funded by an EPSRC DTA
  - Hardware necessary for this project was funded from Jack Kelly's Intel EU PhD Fellowship
- **name**: UK-DALE
- **number_of_buildings**: 5
- **related_documents**:
  - Dataset is available for download from http://www.doc.ic.ac.uk/~dk3810/data/ (Dataset is available for download from http://www.doc.ic.ac.uk/~dk3810/data/)
  - Dataset is also available from the UK Energy Research Council's Energy Data Centre: The 1-second data is available from http://data.ukedc.rl.ac.uk/cgi-bin/dataset_catalogue/view.cgi.py?id=19 and the 6-second data is available from http://data.ukedc.rl.ac.uk/cgi-bin/dataset_catalogue/view.cgi.py?id=18 but please note that this archive is updated less frequently than the data on www.doc.ic.ac.uk/~dk3810/data/ (Dataset is also available from the UK Energy Research Council's Energy Data Centre: The 1-second data is available from http://data.ukedc.rl.ac.uk/cgi-bin/dataset_catalogue/view.cgi.py?id=19 and the 6-second data is available from http://data.ukedc.rl.ac.uk/cgi-bin/dataset_catalogue/view.cgi.py?id=18 but please note that this archive is updated less frequently than the data on www.doc.ic.ac.uk/~dk3810/data/)
  - This research paper describes the data collection: http://arxiv.org/abs/1404.0284 (This research paper describes the data collection: http://arxiv.org/abs/1404.0284)
  - The following poster describes the metering setup and provides some analyses: Jack Kelly and William Knottenbelt. Smart Meter Disaggregation: Data Collection & Analysis. UK Energy Research Council Summer School Ph.D. poster session. June 2013. PDF: http://www.doc.ic.ac.uk/~dk3810/writing/UKERC_poster2013_v2.pdf (The following poster describes the metering setup and provides some analyses: Jack Kelly and William Knottenbelt. Smart Meter Disaggregation: Data Collection & Analysis. UK Energy Research Council Summer School Ph.D. poster session. June 2013. PDF: http://www.doc.ic.ac.uk/~dk3810/writing/UKERC_poster2013_v2.pdf)
- **contact**: jack.kelly@imperial.ac.uk

- **timeframe**:
  - **start**: 2012-11-09T22:28:15+00:00
  - **end**: 2015-01-05T06:26:44+00:00
- **geospatial_coverage**: Southern England
- **creators**:
  - Kelly, Jack
- **schema**: https://github.com/nilmtk/nilm_metadata/tree/v0.2 (https://github.com/nilmtk/nilm_metadata/tree/v0.2)

Print out Buildings

In [3]:
```
print_dict(ukdale.buildings)
```

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

1.3 Print out the sub-metered appliances in each building

In [4]:
```
for build in ukdale.buildings:
    print("Appliances of Building " +str(build))
    print(ukdale.buildings[build].elec.submeters())
    print("---")
```

```
Appliances of Building 1
MeterGroup(meters=
  ElecMeter(instance=2, building=1, dataset='UK-DALE', appliances=
[Appliance(type='boiler', instance=1)])
  ElecMeter(instance=3, building=1, dataset='UK-DALE', appliances=
[Appliance(type='solar thermal pumping station', instance=1)])
  ElecMeter(instance=4, building=1, dataset='UK-DALE', appliances=
[Appliance(type='laptop computer', instance=1), Appliance(type='la
ptop computer', instance=3)])
  ElecMeter(instance=5, building=1, dataset='UK-DALE', appliances=
[Appliance(type='washer dryer', instance=1)])
  ElecMeter(instance=6, building=1, dataset='UK-DALE', appliances=
[Appliance(type='dish washer', instance=1)])
  ElecMeter(instance=7, building=1, dataset='UK-DALE', appliances=
[Appliance(type='television', instance=1)])
  ElecMeter(instance=8, building=1, dataset='UK-DALE', appliances=
[Appliance(type='light', instance=1), Appliance(type='light', inst
ance=2)])
  ElecMeter(instance=9, building=1, dataset='UK-DALE', appliances=
[Appliance(type='HTPC', instance=1)])
  ElecMeter(instance=10, building=1, dataset='UK-DALE', appliances
=[Appliance(type='kettle', instance=1), Appliance(type='food proce
ssor', instance=1), Appliance(type='toasted sandwich maker', insta
nce=1)])
```

```
  ElecMeter(instance=11, building=1, dataset='UK-DALE', appliances
=[Appliance(type='toaster', instance=1), Appliance(type='kitchen a
id', instance=1), Appliance(type='food processor', instance=2)])
  ElecMeter(instance=12, building=1, dataset='UK-DALE', appliances
=[Appliance(type='fridge freezer', instance=1)])
  ElecMeter(instance=13, building=1, dataset='UK-DALE', appliances
=[Appliance(type='microwave', instance=1)])
  ElecMeter(instance=14, building=1, dataset='UK-DALE', appliances
=[Appliance(type='computer monitor', instance=1)])
  ElecMeter(instance=15, building=1, dataset='UK-DALE', appliances
=[Appliance(type='audio system', instance=1)])
  ElecMeter(instance=16, building=1, dataset='UK-DALE', appliances
=[Appliance(type='breadmaker', instance=1)])
  ElecMeter(instance=17, building=1, dataset='UK-DALE', appliances
=[Appliance(type='audio amplifier', instance=1)])
  ElecMeter(instance=18, building=1, dataset='UK-DALE', appliances
=[Appliance(type='broadband router', instance=1)])
  ElecMeter(instance=19, building=1, dataset='UK-DALE', appliances
=[Appliance(type='light', instance=3)])
  ElecMeter(instance=20, building=1, dataset='UK-DALE', appliances
=[Appliance(type='soldering iron', instance=1)])
  ElecMeter(instance=21, building=1, dataset='UK-DALE', appliances
=[Appliance(type='ethernet switch', instance=1), Appliance(type='U
SB hub', instance=1)])
  ElecMeter(instance=22, building=1, dataset='UK-DALE', appliances
=[Appliance(type='vacuum cleaner', instance=1)])
  ElecMeter(instance=23, building=1, dataset='UK-DALE', appliances
=[Appliance(type='light', instance=4)])
  ElecMeter(instance=24, building=1, dataset='UK-DALE', appliances
=[Appliance(type='light', instance=5)])
  ElecMeter(instance=25, building=1, dataset='UK-DALE', appliances
=[Appliance(type='light', instance=16)])
  ElecMeter(instance=26, building=1, dataset='UK-DALE', appliances
=[Appliance(type='light', instance=6)])
  ElecMeter(instance=27, building=1, dataset='UK-DALE', appliances
=[Appliance(type='tablet computer charger', instance=1)])
  ElecMeter(instance=28, building=1, dataset='UK-DALE', appliances
=[Appliance(type='active subwoofer', instance=1)])
  ElecMeter(instance=29, building=1, dataset='UK-DALE', appliances
=[Appliance(type='light', instance=7)])
  ElecMeter(instance=30, building=1, dataset='UK-DALE', appliances
=[Appliance(type='radio', instance=1)])
  ElecMeter(instance=31, building=1, dataset='UK-DALE', appliances
=[Appliance(type='light', instance=8)])
  ElecMeter(instance=32, building=1, dataset='UK-DALE', appliances
=[Appliance(type='wireless phone charger', instance=1), Appliance(
type='audio system', instance=2)])
  ElecMeter(instance=33, building=1, dataset='UK-DALE', appliances
=[Appliance(type='light', instance=9)])
  ElecMeter(instance=34, building=1, dataset='UK-DALE', appliances
=[Appliance(type='mobile phone charger', instance=1)])
  ElecMeter(instance=35, building=1, dataset='UK-DALE', appliances
=[Appliance(type='light', instance=10)])
  ElecMeter(instance=36, building=1, dataset='UK-DALE', appliances
=[Appliance(type='coffee maker', instance=1)])
  ElecMeter(instance=37, building=1, dataset='UK-DALE', appliances
=[Appliance(type='radio', instance=2)])
```

```
  ElecMeter(instance=38, building=1, dataset='UK-DALE', appliances
=[Appliance(type='mobile phone charger', instance=2), Appliance(ty
pe='baby monitor', instance=2), Appliance(type='radio', instance=3
)])
  ElecMeter(instance=39, building=1, dataset='UK-DALE', appliances
=[Appliance(type='hair dryer', instance=1)])
  ElecMeter(instance=40, building=1, dataset='UK-DALE', appliances
=[Appliance(type='hair straighteners', instance=1)])
  ElecMeter(instance=41, building=1, dataset='UK-DALE', appliances
=[Appliance(type='clothes iron', instance=1)])
  ElecMeter(instance=42, building=1, dataset='UK-DALE', appliances
=[Appliance(type='oven', instance=1)])
  ElecMeter(instance=43, building=1, dataset='UK-DALE', appliances
=[Appliance(type='computer', instance=1), Appliance(type='external
hard disk', instance=1)])
  ElecMeter(instance=44, building=1, dataset='UK-DALE', appliances
=[Appliance(type='light', instance=11)])
  ElecMeter(instance=45, building=1, dataset='UK-DALE', appliances
=[Appliance(type='light', instance=12)])
  ElecMeter(instance=46, building=1, dataset='UK-DALE', appliances
=[Appliance(type='baby monitor', instance=1)])
  ElecMeter(instance=47, building=1, dataset='UK-DALE', appliances
=[Appliance(type='charger', instance=1)])
  ElecMeter(instance=48, building=1, dataset='UK-DALE', appliances
=[Appliance(type='light', instance=13)])
  ElecMeter(instance=49, building=1, dataset='UK-DALE', appliances
=[Appliance(type='light', instance=14)])
  ElecMeter(instance=50, building=1, dataset='UK-DALE', appliances
=[Appliance(type='light', instance=15)])
  ElecMeter(instance=51, building=1, dataset='UK-DALE', appliances
=[Appliance(type='desktop computer', instance=1)])
  ElecMeter(instance=52, building=1, dataset='UK-DALE', appliances
=[Appliance(type='fan', instance=1)])
  ElecMeter(instance=53, building=1, dataset='UK-DALE', appliances
=[Appliance(type='printer', instance=1)])
)
---
Appliances of Building 2
MeterGroup(meters=
  ElecMeter(instance=2, building=2, dataset='UK-DALE', appliances=
[Appliance(type='laptop computer', instance=1)])
  ElecMeter(instance=3, building=2, dataset='UK-DALE', appliances=
[Appliance(type='computer monitor', instance=1)])
  ElecMeter(instance=4, building=2, dataset='UK-DALE', appliances=
[Appliance(type='active speaker', instance=1)])
  ElecMeter(instance=5, building=2, dataset='UK-DALE', appliances=
[Appliance(type='computer', instance=1)])
  ElecMeter(instance=6, building=2, dataset='UK-DALE', appliances=
[Appliance(type='broadband router', instance=1)])
  ElecMeter(instance=7, building=2, dataset='UK-DALE', appliances=
[Appliance(type='external hard disk', instance=1)])
  ElecMeter(instance=8, building=2, dataset='UK-DALE', appliances=
[Appliance(type='kettle', instance=1)])
  ElecMeter(instance=9, building=2, dataset='UK-DALE', appliances=
[Appliance(type='rice cooker', instance=1)])
  ElecMeter(instance=10, building=2, dataset='UK-DALE', appliances
=[Appliance(type='running machine', instance=1)])
```

```
  ElecMeter(instance=11, building=2, dataset='UK-DALE', appliances
=[Appliance(type='laptop computer', instance=2)])
  ElecMeter(instance=12, building=2, dataset='UK-DALE', appliances
=[Appliance(type='washing machine', instance=1)])
  ElecMeter(instance=13, building=2, dataset='UK-DALE', appliances
=[Appliance(type='dish washer', instance=1)])
  ElecMeter(instance=14, building=2, dataset='UK-DALE', appliances
=[Appliance(type='fridge', instance=1)])
  ElecMeter(instance=15, building=2, dataset='UK-DALE', appliances
=[Appliance(type='microwave', instance=1)])
  ElecMeter(instance=16, building=2, dataset='UK-DALE', appliances
=[Appliance(type='toaster', instance=1)])
  ElecMeter(instance=17, building=2, dataset='UK-DALE', appliances
=[Appliance(type='games console', instance=1)])
  ElecMeter(instance=18, building=2, dataset='UK-DALE', appliances
=[Appliance(type='modem', instance=1)])
  ElecMeter(instance=19, building=2, dataset='UK-DALE', appliances
=[Appliance(type='cooker', instance=1)])
)
---
Appliances of Building 3
MeterGroup(meters=
  ElecMeter(instance=2, building=3, dataset='UK-DALE', appliances=
[Appliance(type='kettle', instance=1)])
  ElecMeter(instance=3, building=3, dataset='UK-DALE', appliances=
[Appliance(type='electric space heater', instance=1)])
  ElecMeter(instance=4, building=3, dataset='UK-DALE', appliances=
[Appliance(type='laptop computer', instance=1)])
  ElecMeter(instance=5, building=3, dataset='UK-DALE', appliances=
[Appliance(type='projector', instance=1)])
)
---
Appliances of Building 4
MeterGroup(meters=
  ElecMeter(instance=2, building=4, dataset='UK-DALE', appliances=
[Appliance(type='television', instance=1), Appliance(type='DVD pla
yer', instance=1), Appliance(type='set top box', instance=1), Appl
iance(type='light', instance=1)])
  ElecMeter(instance=3, building=4, dataset='UK-DALE', appliances=
[Appliance(type='kettle', instance=1), Appliance(type='radio', ins
tance=1)])
  ElecMeter(instance=4, building=4, dataset='UK-DALE', appliances=
[Appliance(type='boiler', instance=1)])
  ElecMeter(instance=5, building=4, dataset='UK-DALE', appliances=
[Appliance(type='freezer', instance=1)])
  ElecMeter(instance=6, building=4, dataset='UK-DALE', appliances=
[Appliance(type='washing machine', instance=1), Appliance(type='mi
crowave', instance=1), Appliance(type='breadmaker', instance=1)])
)
---
Appliances of Building 5
MeterGroup(meters=
  ElecMeter(instance=2, building=5, dataset='UK-DALE', appliances=
[Appliance(type='active speaker', instance=1)])
  ElecMeter(instance=3, building=5, dataset='UK-DALE', appliances=
[Appliance(type='desktop computer', instance=1)])
  ElecMeter(instance=4, building=5, dataset='UK-DALE', appliances=
```

```
[Appliance(type='hair dryer', instance=1)])
  ElecMeter(instance=5, building=5, dataset='UK-DALE', appliances=
[Appliance(type='television', instance=1)])
  ElecMeter(instance=6, building=5, dataset='UK-DALE', appliances=
[Appliance(type='computer monitor', instance=1)])
  ElecMeter(instance=7, building=5, dataset='UK-DALE', appliances=
[Appliance(type='running machine', instance=1)])
  ElecMeter(instance=8, building=5, dataset='UK-DALE', appliances=
[Appliance(type='network attached storage', instance=1)])
  ElecMeter(instance=9, building=5, dataset='UK-DALE', appliances=
[Appliance(type='server computer', instance=1)])
  ElecMeter(instance=10, building=5, dataset='UK-DALE', appliances
=[Appliance(type='computer monitor', instance=2)])
  ElecMeter(instance=11, building=5, dataset='UK-DALE', appliances
=[Appliance(type='games console', instance=1)])
  ElecMeter(instance=12, building=5, dataset='UK-DALE', appliances
=[Appliance(type='clothes iron', instance=1)])
  ElecMeter(instance=13, building=5, dataset='UK-DALE', appliances
=[Appliance(type='coffee maker', instance=1)])
  ElecMeter(instance=14, building=5, dataset='UK-DALE', appliances
=[Appliance(type='desktop computer', instance=2)])
  ElecMeter(instance=15, building=5, dataset='UK-DALE', appliances
=[Appliance(type='toaster', instance=1)])
  ElecMeter(instance=16, building=5, dataset='UK-DALE', appliances
=[Appliance(type='audio amplifier', instance=1)])
  ElecMeter(instance=17, building=5, dataset='UK-DALE', appliances
=[Appliance(type='set top box', instance=1)])
  ElecMeter(instance=18, building=5, dataset='UK-DALE', appliances
=[Appliance(type='kettle', instance=1)])
  ElecMeter(instance=19, building=5, dataset='UK-DALE', appliances
=[Appliance(type='fridge freezer', instance=1)])
  ElecMeter(instance=20, building=5, dataset='UK-DALE', appliances
=[Appliance(type='electric oven', instance=1)])
  ElecMeter(instance=21, building=5, dataset='UK-DALE', appliances
=[Appliance(type='electric stove', instance=1)])
  ElecMeter(instance=22, building=5, dataset='UK-DALE', appliances
=[Appliance(type='dish washer', instance=1)])
  ElecMeter(instance=23, building=5, dataset='UK-DALE', appliances
=[Appliance(type='microwave', instance=1)])
  ElecMeter(instance=24, building=5, dataset='UK-DALE', appliances
=[Appliance(type='washer dryer', instance=1)])
  ElecMeter(instance=25, building=5, dataset='UK-DALE', appliances
=[Appliance(type='vacuum cleaner', instance=1)])
)
---
```

In [5]:
```
elec = ukdale.buildings[1].elec
```

1.4 Calculate the total energy consumption for building 1 in kWh

In [6]:
```
elec.mains().total_energy()
```

Out[6]:
```
apparent    5835.953591
active      5008.108254
dtype: float64
```

1.5 Print out the type of power for mains and sub-meters

In [7]:
```
elec.mains().available_ac_types('power')
```

Out[7]:
```
['active', 'apparent']
```

In [8]:
```
elec.submeters().available_ac_types('power')
```

Out[8]:
```
['active', 'apparent']
```

Exercise 2

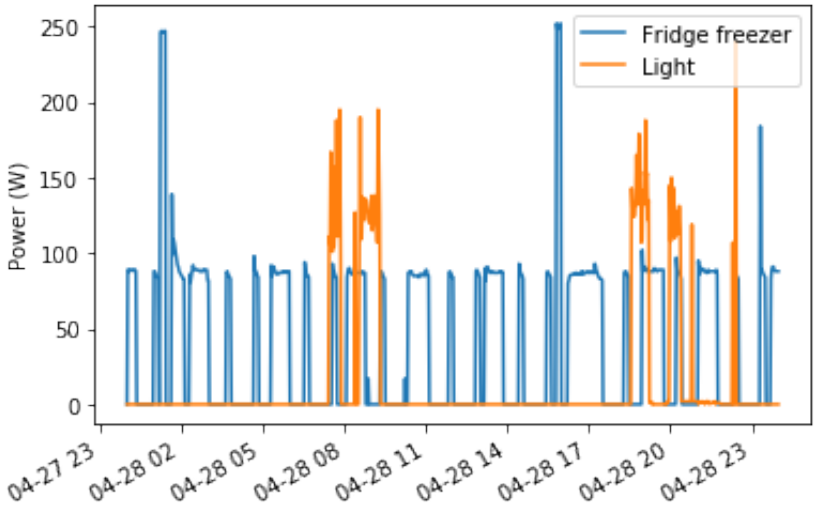
2.1 Timeframed "Fridge Freezer" and "Light" Power Plot

In [9]:
```
ukdale_window = DataSet('./data/ukdale.h5')
ukdale_window.set_window(start='2014-04-28', end='2014-04-29')

fridge_meter = ukdale_window.buildings[1].elec['fridge freezer']
light_meter = ukdale_window.buildings[1].elec['light']
elec = ukdale_window.buildings[1].elec
```

In [10]:
```
fridge_meter.plot()
light_meter.plot()
```

```
/Users/hardanimaulana/anaconda2/envs/nilmtk-env/lib/python3.6/site
-packages/nilmtk/elecmeter.py:189: RuntimeWarning: Multiple applia
nces are associated with meter {} but none are marked as the domin
ant appliance. Hence returning the first appliance in the list.
  ' returning the first appliance in the list.', RuntimeWarning)
```

Out[10]:
```
<matplotlib.axes._subplots.AxesSubplot at 0xa1b6f2f60>
```



2.2 Plot Overall Consumption For that time period

In [11]:
```
all_window = next(ukdale_window.buildings[1].elec.load())
all_window.head()
```

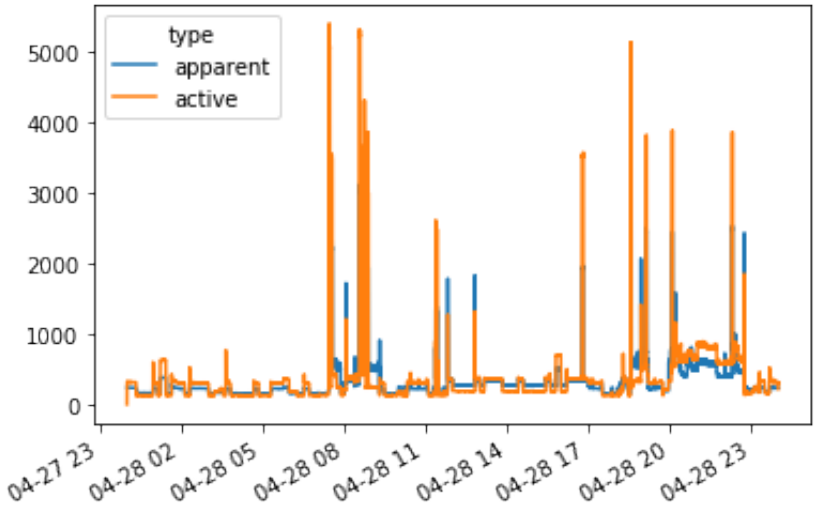
```
Loading data for meter ElecMeterID(instance=54, building=1, datase
t='UK-DALE')
Done loading data all meters for this chunk.
```

Out[11]:

physical_quantity	power		voltage
type	apparent	active	
2014-04-28 00:00:00+01:00	NaN	4.000000	NaN
2014-04-28 00:00:06+01:00	241.339996	229.830002	241.539993
2014-04-28 00:00:12+01:00	241.289993	321.580017	241.460007
2014-04-28 00:00:18+01:00	241.220001	321.570007	241.570007
2014-04-28 00:00:24+01:00	241.529999	322.640015	241.669998

In [12]:
```
all_window['power'].plot()
```

Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0xa1bf33710>

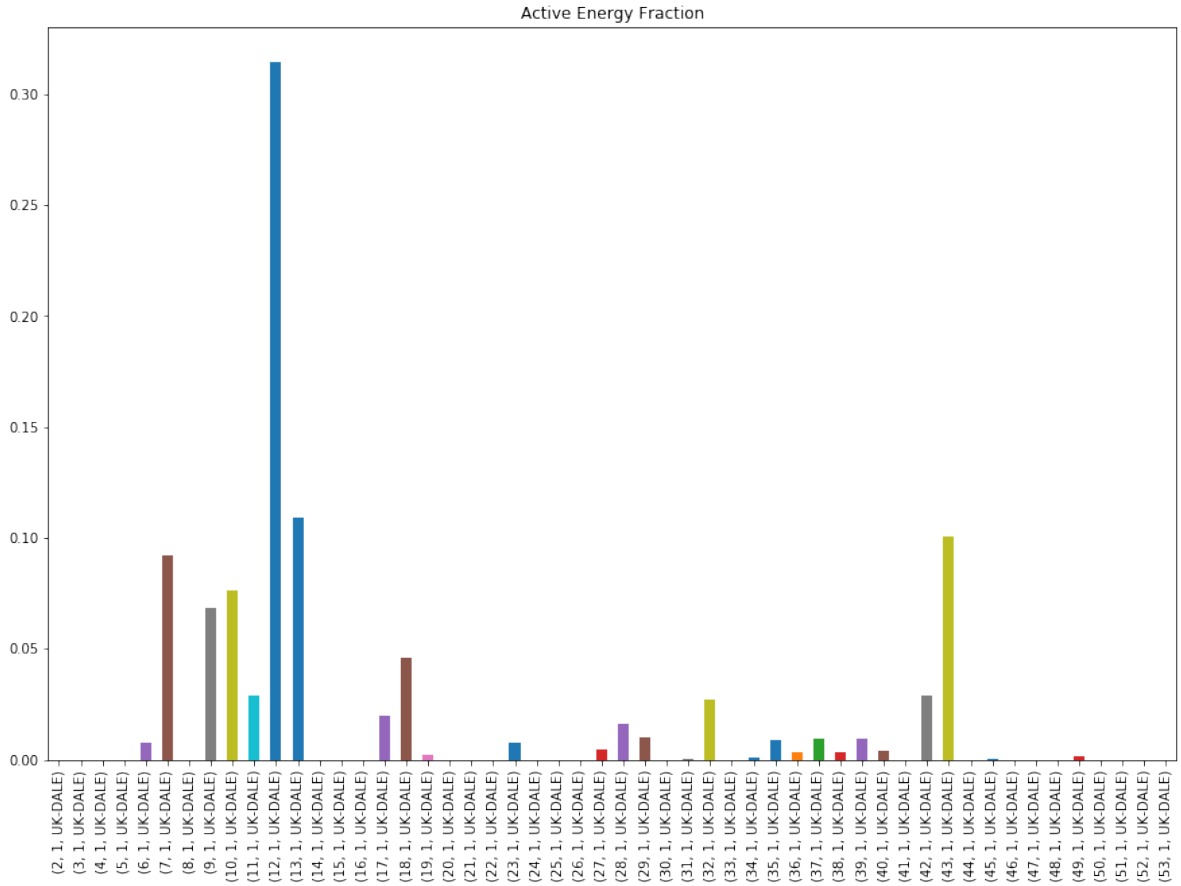


2.3 Calculate and plot the energy consumption fraction for each sub-meter

In [13]:
```
energy_fraction_per_submeter = elec.submeters().energy_per_meter().
transpose().fillna(0)
del energy_fraction_per_submeter['reactive']
active_en = energy_fraction_per_submeter['active']
active_en_frac = active_en/active_en.sum()
active_en_frac.plot(kind="bar", figsize=(15,10), title="Active Ener
gy Fraction")
```

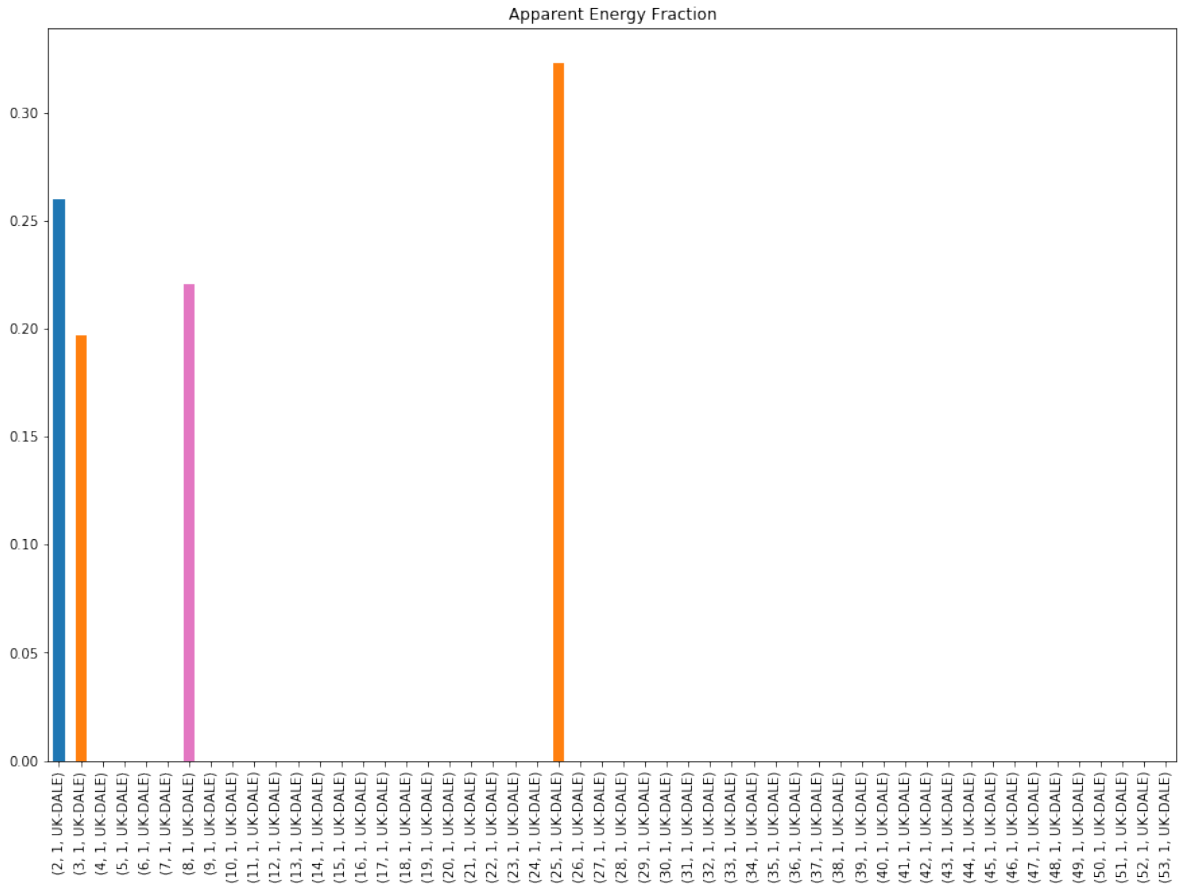
```
52/52 ElecMeter(instance=53, building=1, dataset='UK-DALE', applia
nces=[Appliance(type='printer', instance=1)])ance=1)])e(type='exte
rnal hard disk', instance=1)])e=2), Appliance(type='radio', instan
ce=3)])1)])
```

Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0xa1c658e10>



In [14]:
```
apparent_en = energy_fraction_per_submeter['apparent']
apparent_en_frac = apparent_en/apparent_en.sum()
apparent_en_frac.plot(kind="bar", figsize=(15,10), title="Apparent
Energy Fraction")
```

Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0xa1c234a58>



2.4 Highest Power Consuming Appliance

In [15]:
```
max_appliance = elec.submeters().select_top_k(k=1).energy_per_meter
()
```

```
1/1 ElecMeter(instance=12, building=1, dataset='UK-DALE', applianc
es=[Appliance(type='fridge freezer', instance=1)])1)])e(type='exte
rnal hard disk', instance=1)])e=2), Appliance(type='radio', instan
ce=3)])1)])
```

2.5 Find appliances of the type “single-phase induction motor”

In [16]:
```
elec.select_using_appliances(category='single-phase induction motor
')
```

Out[16]:
```
MeterGroup(meters=
  ElecMeter(instance=2, building=1, dataset='UK-DALE', appliances=
[Appliance(type='boiler', instance=1)])
  ElecMeter(instance=3, building=1, dataset='UK-DALE', appliances=
[Appliance(type='solar thermal pumping station', instance=1)])
  ElecMeter(instance=5, building=1, dataset='UK-DALE', appliances=
[Appliance(type='washer dryer', instance=1)])
  ElecMeter(instance=6, building=1, dataset='UK-DALE', appliances=
[Appliance(type='dish washer', instance=1)])
  ElecMeter(instance=10, building=1, dataset='UK-DALE', appliances
=[Appliance(type='kettle', instance=1), Appliance(type='food proce
ssor', instance=1), Appliance(type='toasted sandwich maker', insta
nce=1)])
  ElecMeter(instance=11, building=1, dataset='UK-DALE', appliances
=[Appliance(type='toaster', instance=1), Appliance(type='kitchen a
id', instance=1), Appliance(type='food processor', instance=2)])
  ElecMeter(instance=12, building=1, dataset='UK-DALE', appliances
=[Appliance(type='fridge freezer', instance=1)])
  ElecMeter(instance=16, building=1, dataset='UK-DALE', appliances
=[Appliance(type='breadmaker', instance=1)])
  ElecMeter(instance=22, building=1, dataset='UK-DALE', appliances
=[Appliance(type='vacuum cleaner', instance=1)])
  ElecMeter(instance=52, building=1, dataset='UK-DALE', appliances
=[Appliance(type='fan', instance=1)])
  ElecMeter(instance=54, building=1, dataset='UK-DALE', site_meter
, appliances=[Appliance(type='immersion heater', instance=1), Appl
iance(type='water pump', instance=1), Appliance(type='security ala
rm', instance=1), Appliance(type='fan', instance=2), Appliance(typ
e='drill', instance=1), Appliance(type='laptop computer', instance
=2)])
)
```

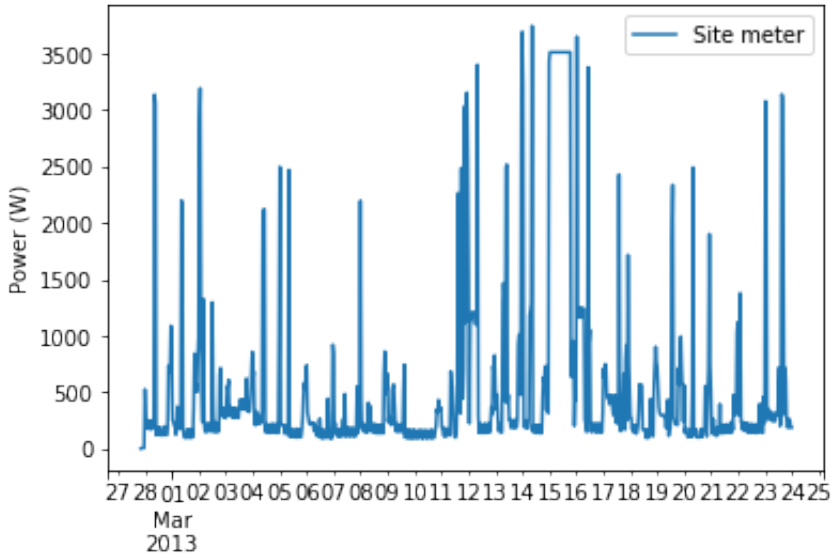
Exercise 3

In [17]:
```
import time
from six import iteritems
import matplotlib.pyplot as plt
import numpy as np

train = DataSet('./data/ukdale.h5')
test = DataSet('./data/ukdale.h5')
```

In [18]:
```
train.set_window(end="24-3-2013")
test.set_window(start="25-3-2013")

train_elec = train.buildings[3].elec
test_elec = test.buildings[3].elec
```

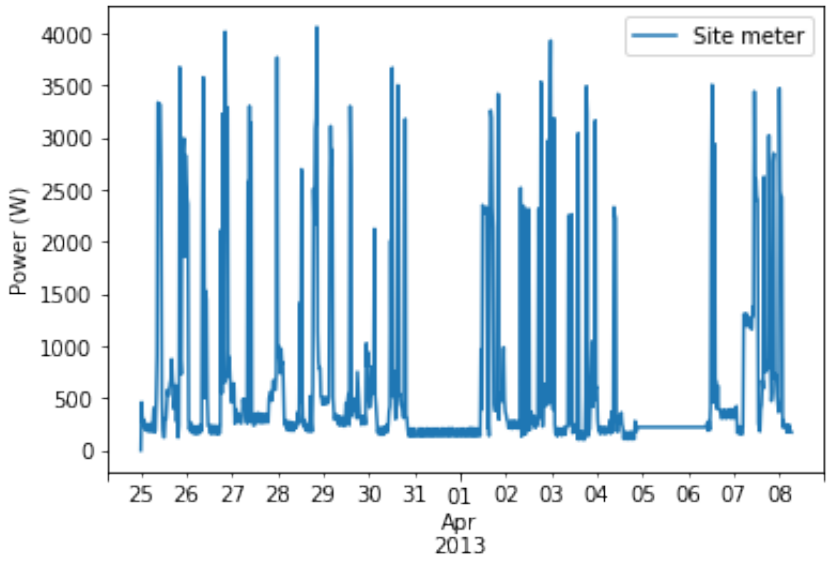
In [19]:
```
train_elec.mains().plot()
```

Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0xa1cd58cf8>



In [20]:
```
test_elec.mains().plot()
```

Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0xa1d7916d8>



In [21]:
```
mains = train_elec.mains()
mains_df = next(mains.load())
mains_df.head()
```

Out[21]:

physical_quantity	power
type	apparent
2013-02-27 20:35:14+00:00	5.0
2013-02-27 20:35:20+00:00	4.0
2013-02-27 20:35:26+00:00	5.0
2013-02-27 20:35:32+00:00	5.0
2013-02-27 20:35:38+00:00	4.0

Prepare a Method for Predicting and Calculating the F-Score

In [22]:
```
from nilmtk.disaggregate import CombinatorialOptimisation, FHMM
from nilmtk.tests.testingtools import data_dir
from sklearn.metrics import f1_score
```

Disaggregate and Calculate the F-Score With this function

In [23]:
```
def disaggr_and_fscore(algorithm,train_elec, test_elec,train_timezo
ne,show_debug=True):
    start = time.time()
    algorithm.train(train_elec,sample_period=6)
    end = time.time()
    print("Train runtime =", end-start, "seconds.")
    pred = {}
    gt= {}

    for i, chunk in enumerate(test_elec.mains().load(sample_period=
6)):
        chunk_drop_na = (chunk).dropna()
        pred[i] = algorithm.disaggregate_chunk(chunk_drop_na)
        gt[i]={}

        for meter in test_elec.submeters().meters:
            # Only use the meters that we trained on (this saves ti
me!)
            gt[i][meter] = next(meter.load(sample_period=6))
        gt[i] = pd.DataFrame({k:v.squeeze() for k,v in iteritems(gt
[i])}, index=next(iter(gt[i].values())).index).dropna()

    gt_overall = pd.concat(gt)
    gt_overall.index = gt_overall.index.droplevel()
    pred_overall = pd.concat(pred)
    pred_overall.index = pred_overall.index.droplevel()
```

```
    gt_overall = gt_overall[pred_overall.columns]

    gt_index_utc = gt_overall.index.tz_convert("UTC")
    pred_index_utc = pred_overall.index.tz_convert("UTC")
    common_index_utc = gt_index_utc.intersection(pred_index_utc)


    common_index_local = common_index_utc.tz_convert(train_timezone
)

    gt_overall = gt_overall.ix[common_index_local]
    pred_overall = pred_overall.ix[common_index_local]

    if show_debug:
        gt_overall.head()

    appliance_labels = [m.label() for m in gt_overall.columns.value
s]
    gt_overall.columns = appliance_labels
    pred_overall.columns = appliance_labels

    if show_debug:
        pred_overall.head()
        pred_overall.head(100000).plot(title="Pred",figsize=(15,5))
        gt_overall.head(100000).plot(title="GT",figsize=(15,5))
        plt.legend()

    resulting_f_score = {}
    threshold_w = 5
    for appliance in gt_overall.columns:
        temp_gt = gt_overall[appliance].copy()
        temp_gt[temp_gt<=threshold_w] = 0
        temp_gt[temp_gt>threshold_w] = 1
        temp_pred = pred_overall[appliance].copy()
        temp_pred[temp_pred<=threshold_w] = 0
        temp_pred[temp_pred>threshold_w] = 1
        resulting_f_score[appliance] = f1_score(temp_gt, temp_pred)

    return resulting_f_score
```

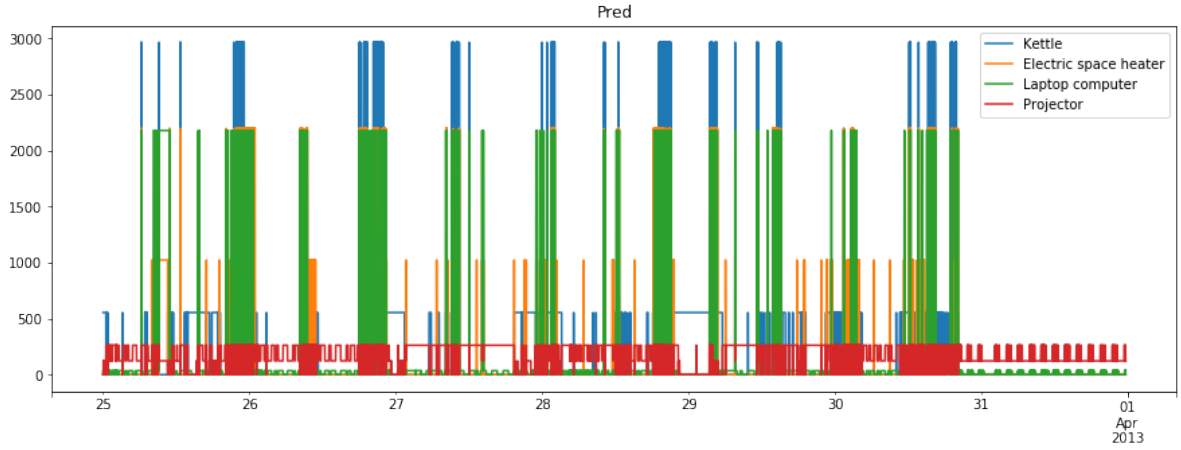
In [24]:
```
classifiers = {'CO':CombinatorialOptimisation(), 'FHMM':FHMM()}
resulting_f_scores = {}
```

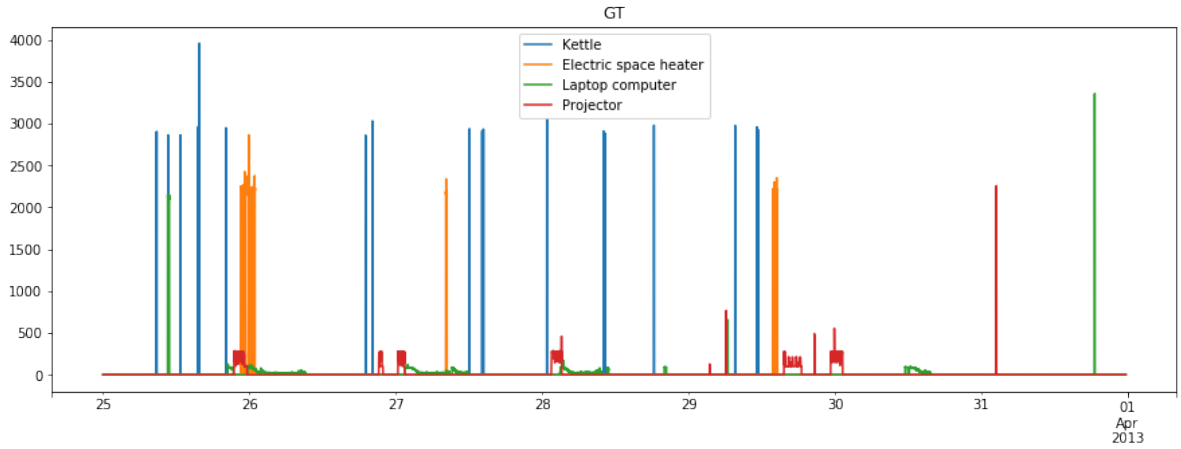
3.1 Combinatorial Optimisation

In [25]:
```
resulting_f_scores['CO'] = disaggr_and_fscore(CombinatorialOptimisa
tion(),train_elec, test_elec,train.metadata['timezone'],show_debug=
True)
```

```
Training model for submeter 'ElecMeter(instance=2, building=3, dat
aset='UK-DALE', appliances=[Appliance(type='kettle', instance=1)])
'
```

```
/Users/hardanimaulana/anaconda2/envs/nilmtk-env/lib/python3.6/site
-packages/sklearn/metrics/pairwise.py:257: RuntimeWarning: invalid
value encountered in sqrt
  return distances if squared else np.sqrt(distances, out=distance
s)

Training model for submeter 'ElecMeter(instance=3, building=3, dat
aset='UK-DALE', appliances=[Appliance(type='electric space heater'
, instance=1)])'

/Users/hardanimaulana/anaconda2/envs/nilmtk-env/lib/python3.6/site
-packages/sklearn/metrics/pairwise.py:257: RuntimeWarning: invalid
value encountered in sqrt
  return distances if squared else np.sqrt(distances, out=distance
s)

Training model for submeter 'ElecMeter(instance=4, building=3, dat
aset='UK-DALE', appliances=[Appliance(type='laptop computer', inst
ance=1)])'

/Users/hardanimaulana/anaconda2/envs/nilmtk-env/lib/python3.6/site
-packages/sklearn/metrics/pairwise.py:257: RuntimeWarning: invalid
value encountered in sqrt
  return distances if squared else np.sqrt(distances, out=distance
s)

Training model for submeter 'ElecMeter(instance=5, building=3, dat
aset='UK-DALE', appliances=[Appliance(type='projector', instance=1
)])'

/Users/hardanimaulana/anaconda2/envs/nilmtk-env/lib/python3.6/site
-packages/sklearn/metrics/pairwise.py:257: RuntimeWarning: invalid
value encountered in sqrt
  return distances if squared else np.sqrt(distances, out=distance
s)

Done training!
Train runtime = 2.218951940536499 seconds.
Estimating power demand for 'ElecMeter(instance=2, building=3, dat
aset='UK-DALE', appliances=[Appliance(type='kettle', instance=1)])
'
Estimating power demand for 'ElecMeter(instance=3, building=3, dat
aset='UK-DALE', appliances=[Appliance(type='electric space heater'
, instance=1)])'
Estimating power demand for 'ElecMeter(instance=4, building=3, dat
aset='UK-DALE', appliances=[Appliance(type='laptop computer', inst
ance=1)])'
Estimating power demand for 'ElecMeter(instance=5, building=3, dat
aset='UK-DALE', appliances=[Appliance(type='projector', instance=1
)])'
```



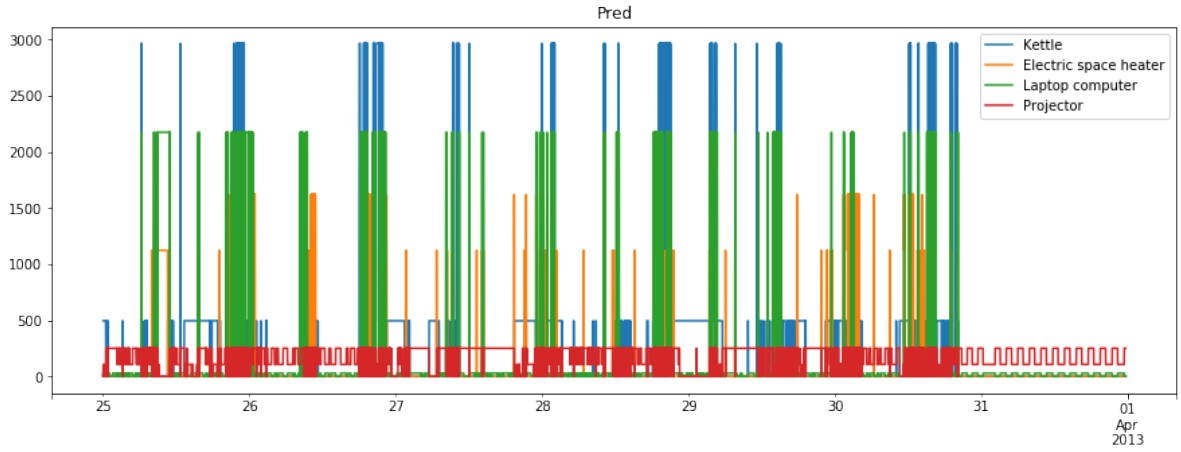


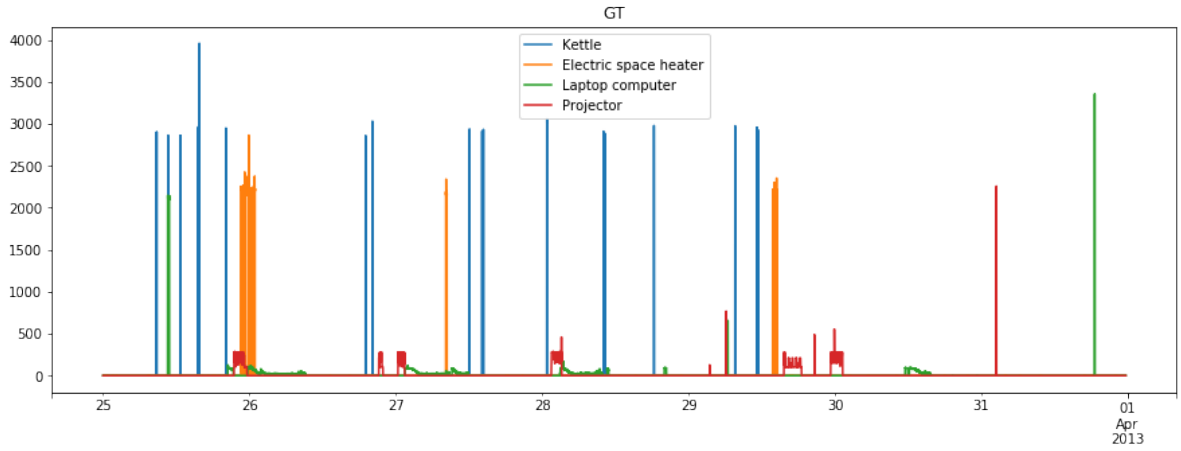
3.2 FHMM

In [26]:
```
resulting_f_scores['FHMM'] = disaggr_and_fscore(FHMM(),train_elec,
test_elec,train.metadata['timezone'],show_debug=True)
```

```
/Users/hardanimaulana/anaconda2/envs/nilmtk-env/lib/python3.6/site
-packages/sklearn/metrics/pairwise.py:257: RuntimeWarning: invalid
value encountered in sqrt
  return distances if squared else np.sqrt(distances, out=distance
s)

Training model for submeter 'ElecMeter(instance=2, building=3, dat
aset='UK-DALE', appliances=[Appliance(type='kettle', instance=1)])
'

/Users/hardanimaulana/anaconda2/envs/nilmtk-env/lib/python3.6/site
-packages/sklearn/metrics/pairwise.py:257: RuntimeWarning: invalid
value encountered in sqrt
  return distances if squared else np.sqrt(distances, out=distance
s)

Training model for submeter 'ElecMeter(instance=3, building=3, dat
aset='UK-DALE', appliances=[Appliance(type='electric space heater'
, instance=1)])'
```

```
/Users/hardanimaulana/anaconda2/envs/nilmtk-env/lib/python3.6/site
-packages/sklearn/metrics/pairwise.py:257: RuntimeWarning: invalid
value encountered in sqrt
  return distances if squared else np.sqrt(distances, out=distance
s)

Training model for submeter 'ElecMeter(instance=4, building=3, dat
aset='UK-DALE', appliances=[Appliance(type='laptop computer', inst
ance=1)])'

/Users/hardanimaulana/anaconda2/envs/nilmtk-env/lib/python3.6/site
-packages/sklearn/metrics/pairwise.py:257: RuntimeWarning: invalid
value encountered in sqrt
  return distances if squared else np.sqrt(distances, out=distance
s)

Training model for submeter 'ElecMeter(instance=5, building=3, dat
aset='UK-DALE', appliances=[Appliance(type='projector', instance=1
)])'
Train runtime = 35.651565074920654 seconds.
```





Exercise 4

Compare F-Score of CO and FHMM

In [27]:
```
f_score_df={}
f_score_df['FHMM']=pd.Series(resulting_f_scores['FHMM'])
f_score_df['CO'] = pd.Series(resulting_f_scores['CO'])
f_score_df = pd.DataFrame(f_score_df)
f_score_df
```

Out[27]:

	CO	FHMM
Electric space heater	0.381523	0.367336
Kettle	0.016292	0.009945
Laptop computer	0.374046	0.351022
Projector	0.088006	0.091067

In [28]:
```
f_score_df.plot(kind='bar')
```

Out[28]: `<matplotlib.axes._subplots.AxesSubplot at 0x1a371f8b70>`