**Appliance Event Detection**

**Objective**

Identification of events in an aggregated stream of power consumption from a household. Events here are described as fluctuations in the aggregated data stream caused by a change in the state of an appliance (on/off).

**Data Source**

Onset Data, Device Metadata

**Features**

Onset Data : Mains\_Power, Logged\_Time\_Local

Device Metadata: Device\_ID, Device\_Name, Appliance\_Name, Mean\_Consumption, Device\_Cat

**Algorithm**

**Step 1**

Calculate the following features to track deviations in the stream for some point i:

**Before\_Mean**: Mean of stream n points before i

**After\_Mean**: Mean of stream n points after i

**Per\_Change**: (After\_Mean-Before\_Mean)/Before\_Mean

**Mean\_Difference**: After\_Mean – Before\_mean

**Before\_Max**: Peak reached in the stream between i and (i-n)

**After\_Max**: Peak reached in the stream between i and (i+n)

**Before\_Mean\_10**: Mean of stream n\*f points before i

**After\_Mean\_10**: Mean of stream n\*f points after i

\*\* n are the number of points before/after I, f is a factor to characterize the stream beyond the point n

**Step 2**

Define Transition – Deviations in the stream of data

Any point I will have Transition = True if the following conditions are met:

Mean\_Difference>.01 KW and Mains\_Power <.1 KW

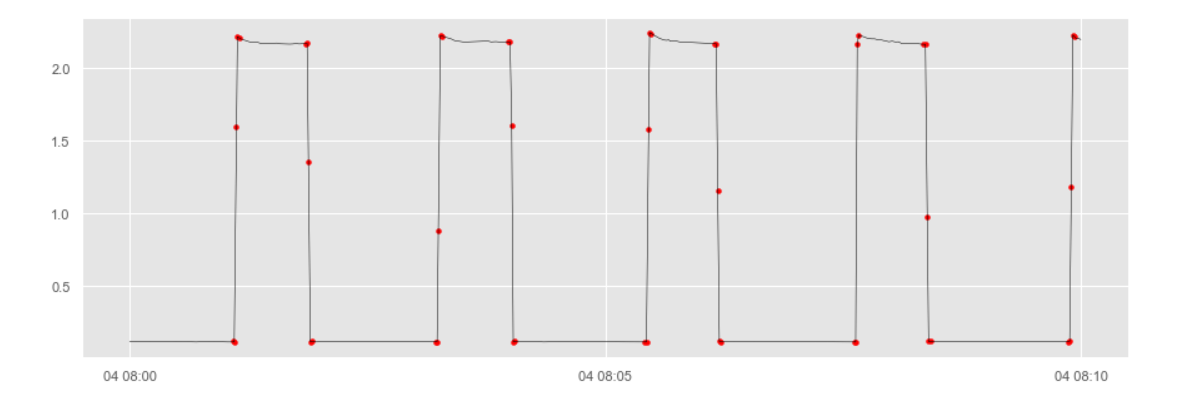
OR

Percentage\_Change>10 and Mains\_Power >.1 KW –(2)

Percentage\_change as a function of Mains\_Power \*\*\*\*\*

Avg Fluctuations for a high powered Device\*\*\*\*\* -- max(per\_change>10, or this)-(2)

Fig 1



As seen in Fig 1, these conditions help in identifying deviations in the data stream but highlights intermediary points as well

**Step 3**

Define transition states:

Transition State = (0,1,2,3)

Transition\_Start=1, Transition= 2, Transition\_End=3, Steady\_State=0

Algorithm:

1. If Transition== True then Transition\_State=2 else Transition\_State=0
2. Iterate through the stream and define a variable to capture the previous stream

Identify points where the transition has started and mark it 1 and mark it 3 whenever the transition ends, Step1 of this algorithm will ensure that all the points in between 1 and 3 are marked with transition state=2. Loop for this algorithm is given below

#### Check with code

Is\_prev\_event =0 (dummy variable)

for i in range(len(sample\_data)):

if Transition == 0: (Assert : Not an Event)

if Is\_prev\_event ==1:

Transition\_State =3 (Asset: Event Ended)

Is\_prev\_event =0

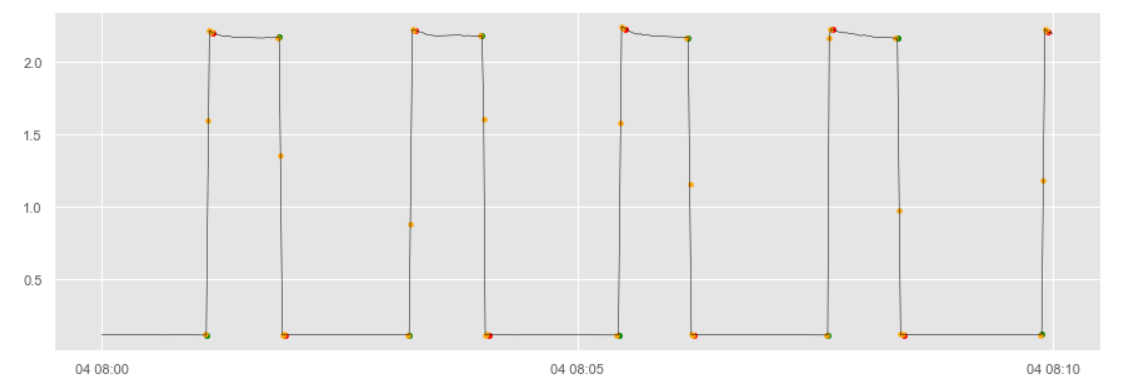
else: (Assert: It is an Event)

if Is\_prev\_event ==0:

Transition=1 (Asset: Event started)

Is\_prev\_event=1

**Fig 2**



**Fig 2.1**

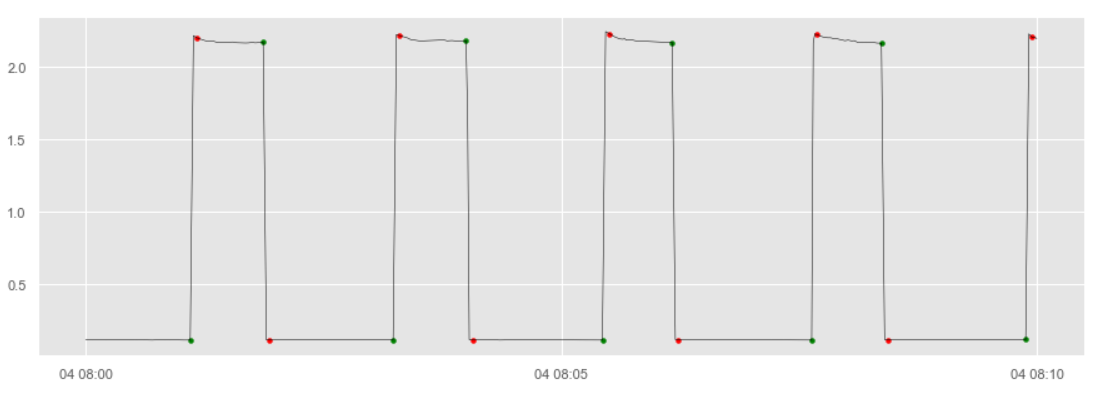
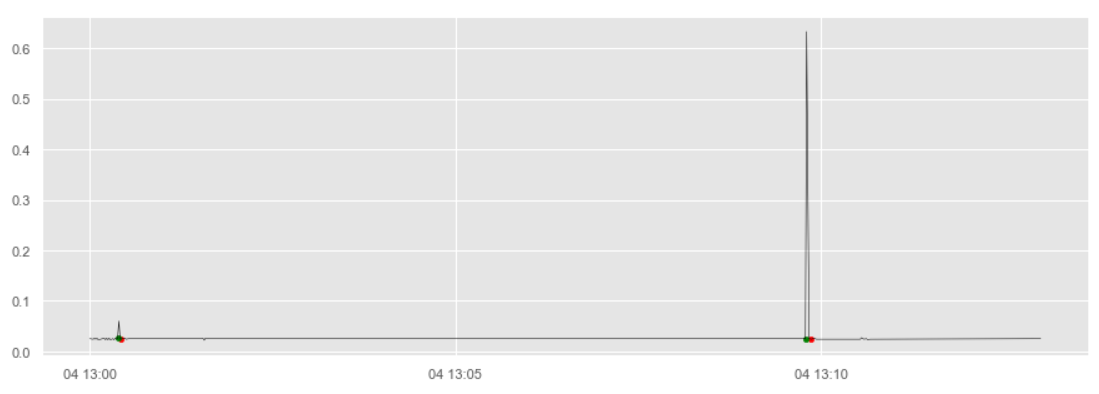


Fig 2 shows all the transition points Orange is Transition\_State(TS) =2, Red is TS=3 and Green is TS=1

Many superimposed points can be seen but when we Only capture the Transition start and End points as in Fig 2.1, we can see a lot of improvement.

Although this algorithm fares well, the downside is that it treats anomalies as events(Fig 2.2)

**Fig 2.2**



**Step 4 :**

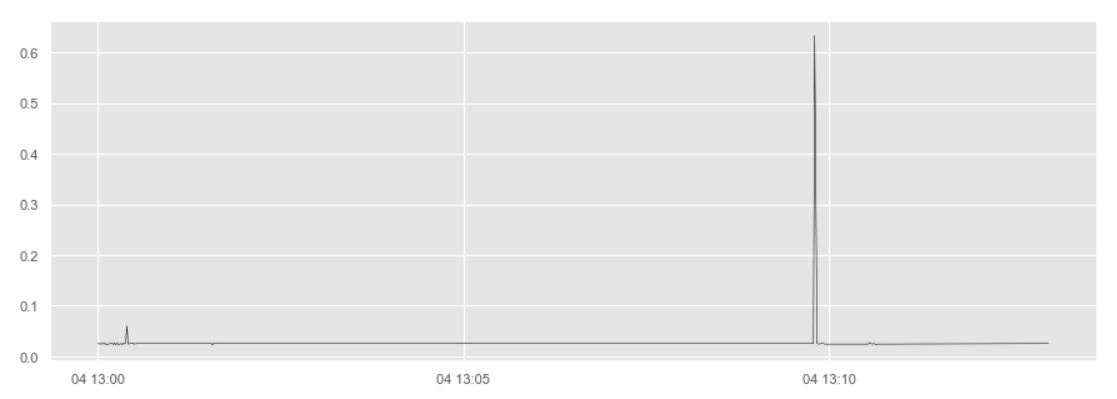
To handle this, Event\_Effect is defined:

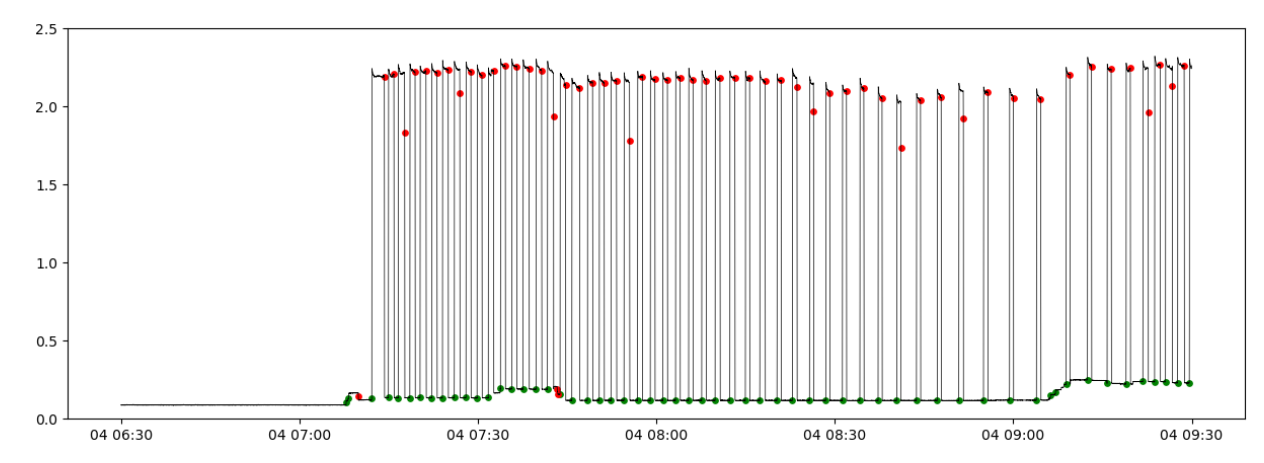
Event\_Effect = |After\_Mean\_10 – Before\_Mean\_10|

Finally Event=TRUE, if at point i , event\_effect >.010 KW

This means that the average increase in the value of the stream due to an event, which handles the above issue as seen in fig 3

**Fig 3**

**Output of Algorithm**



**Post Appliance Event Detection, we move towards Energy Disaggregation and the Final Output of our Analysis.** The above analysis was helpful in identifying Appliance activities. The next task is to build a pipeline to identify the appliance responsible for these events, estimate the energy consumption by each of the appliances and eventually give a statistical report for the same(consumer app).

For this, Two placeholder functions namely:

**Appliance\_Classification()**

**Appliance\_Energy\_Estimation()**

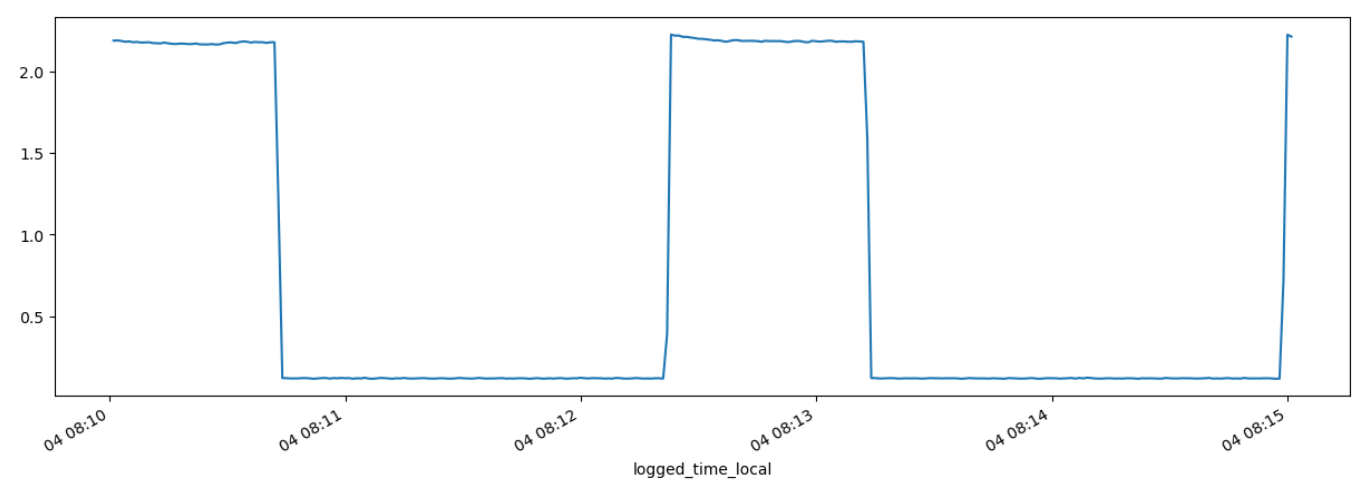
Are created to produce a baseline model and will require future work to increase the accuracy and efficiency of the system. The need for this is shown in the next algorithm of this analysis.

**Algorithm**

The algorithm aims at publishing an output of energy consumption of each appliance in a time window and publishing results based on room/device category/appliance name

Ex. Between the time window 7:10 and 7:15, 3 appliances were switched on





**Global Variables**

**Running Average**: Takes into account the very low energy consuming appliances ex. Router that are running in the background. FLOAT VALUE

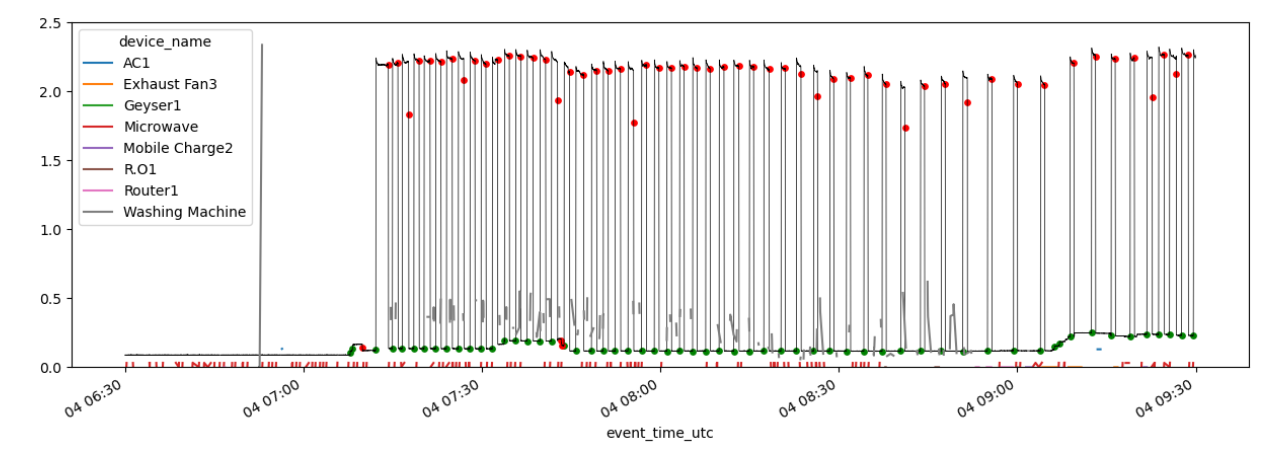
**Active Appliances:** Keeps a track of all the appliances that were in active state in the previous time windows and the time that appliance was active previously. DICTIONARY {Appliance: Start\_Time}

For every n minutes:

1. Create event\_df with columns=['start\_date','end\_date','running\_avg','appliance','status','total\_energy\_consumed','start\_time','end\_time']
2. Create a row in event\_df with appliance=’all’ , total power= sum of all readings/3600, running avg = prev running avg, start time, end time
3. For all the previously active\_appliances create new rows in event\_df with start\_time = initial timestamp of data obtained, appliance name=appliance
4. Iterate through the data obtained in these n minutes
5. If event=1 (On event), run the appliance classification algorithm : **Appliance\_Classification()**  to obtain the appliance which caused the event. Create a new row for this appliance if not there in event\_df of this window, else add new start time with a ‘,’ separator to the start\_time column and update the Active\_Appliance dictionary with this new time and make status=’Active’
6. If event=2 (Off event), run the **Appliance\_Classification** amd the appliance energy estimation algorithm: **Appliance\_Energy\_Estimation()** with parameters from the dictionary- start time, end time which is the current timestamp and the appliance name obtained from the classification Algorithmm, once the energy estimation is completed, we updated the total energy consumed by the particular appliance(by adding it to the previous value if entry already exists). Update the status=’off’ and delete the entry from active\_appliance dict
7. Post the loop, add up energy for each appliance consumed during this time window and update the running average if no other appliance is running, if it is running, use the previous running average.
8. Add ‘other’ row in appliances to get energy consumed from the running average

**INPUT**

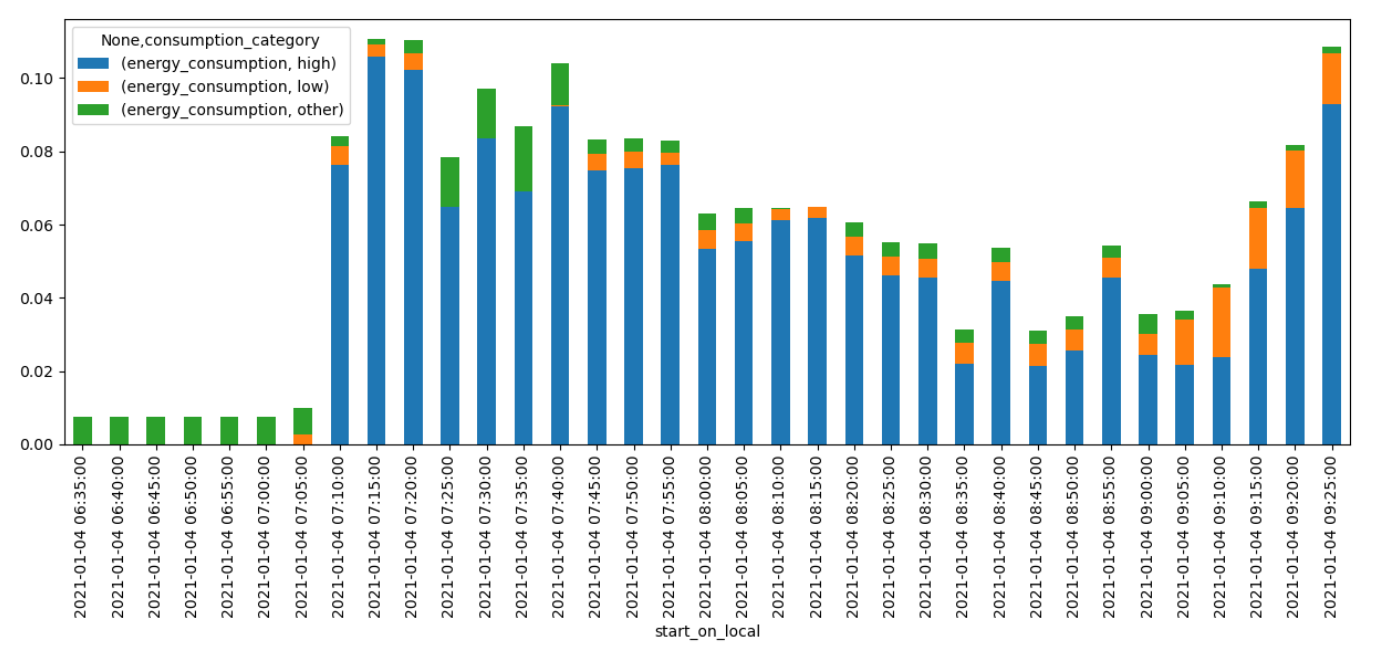
**Original Onset + Tuya Data + Highlighted Events**



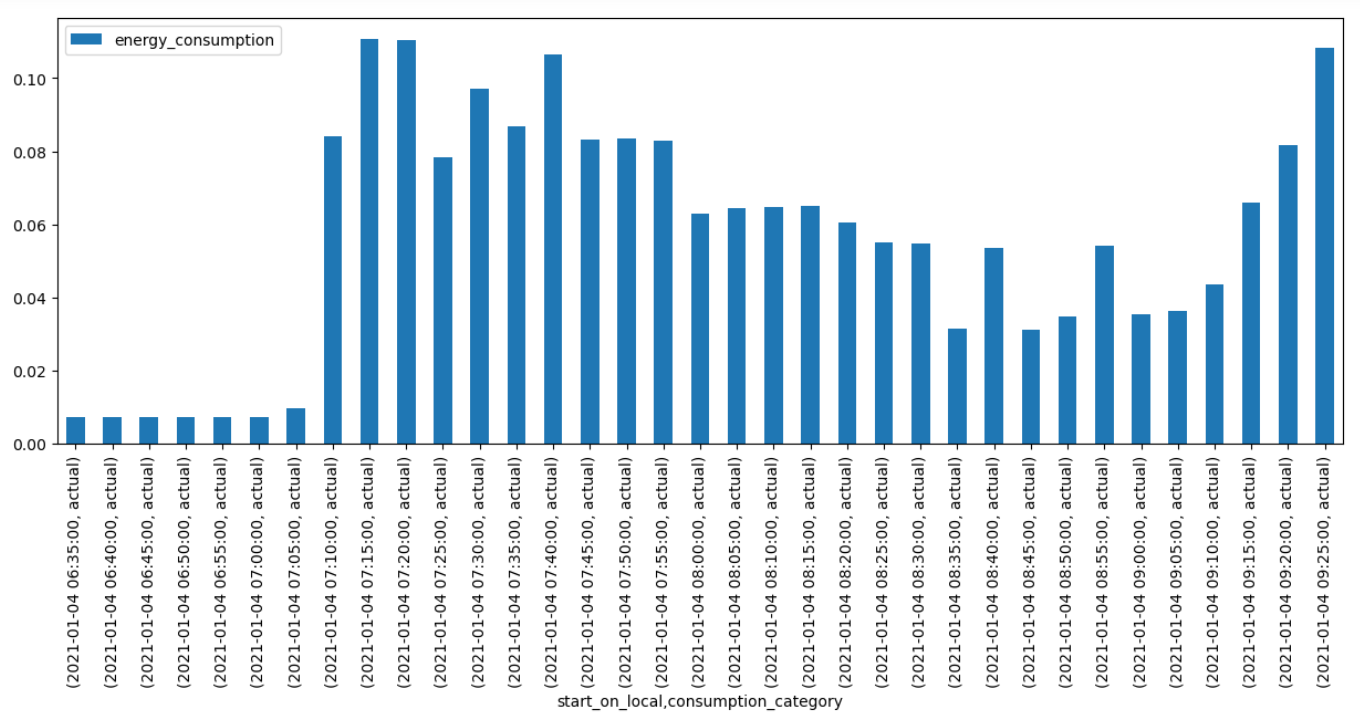
**OUTPUT**

Find below a comparison of the actual vs estimated data

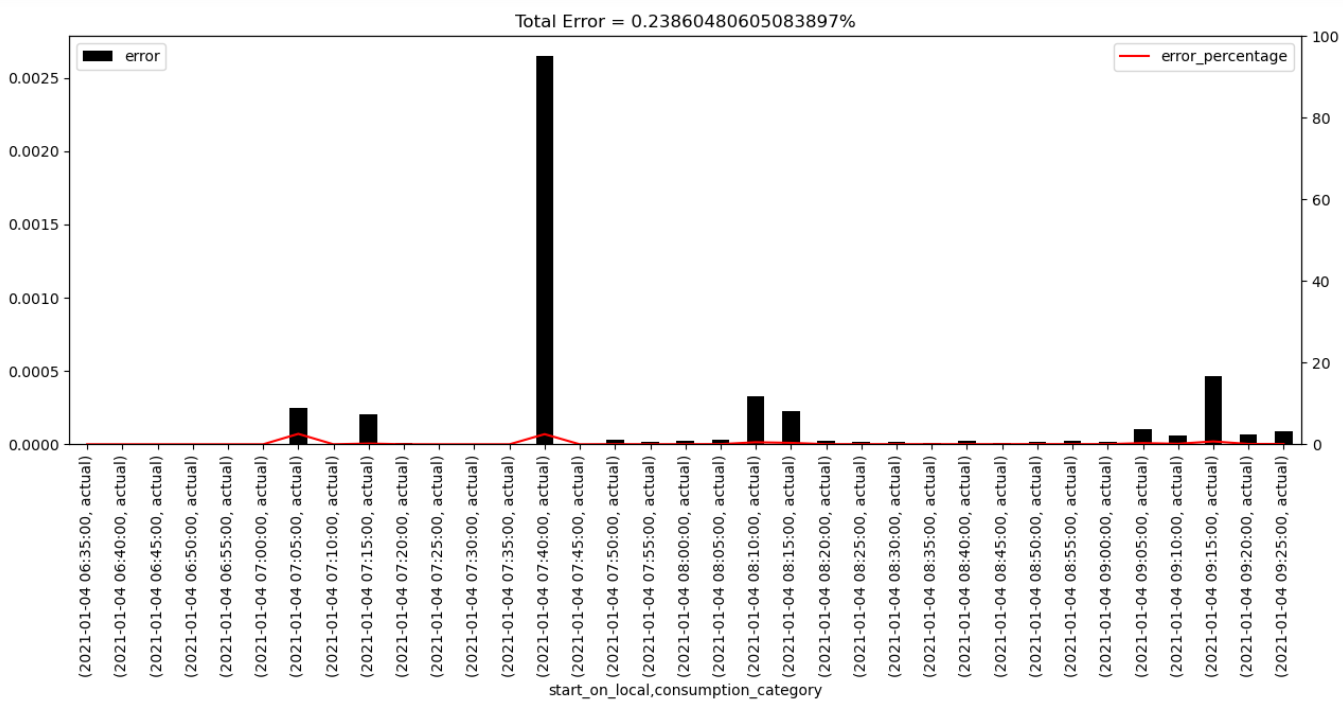
**Estimated(based on device category : High/Medium/Low)**



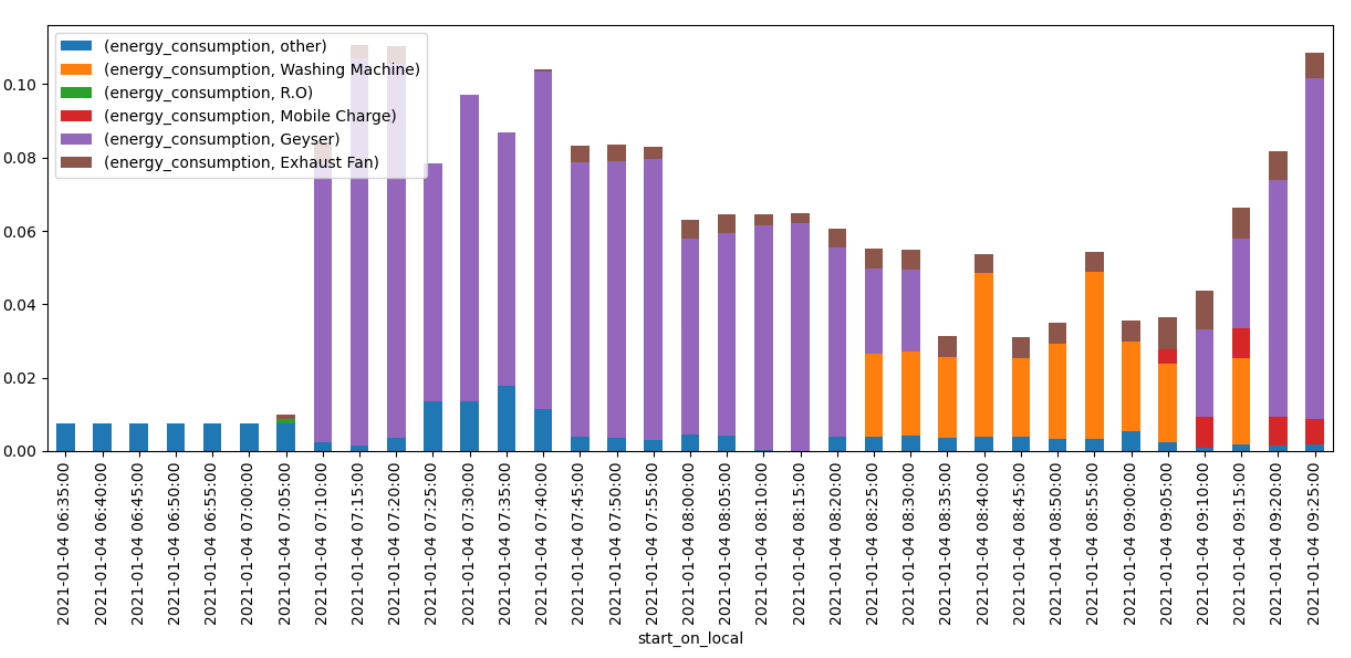
**Actual**



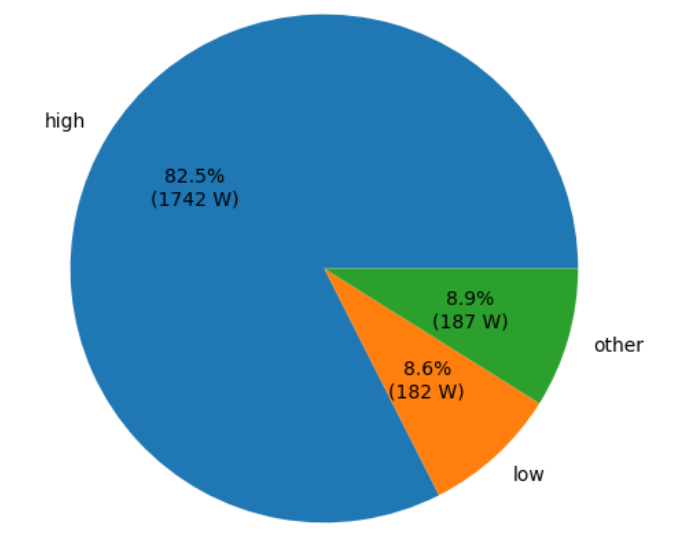
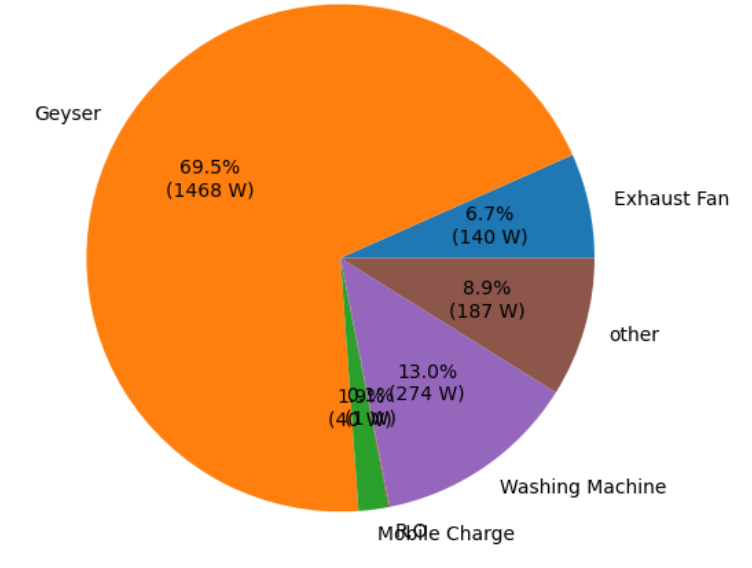
**Actual-Estimated**

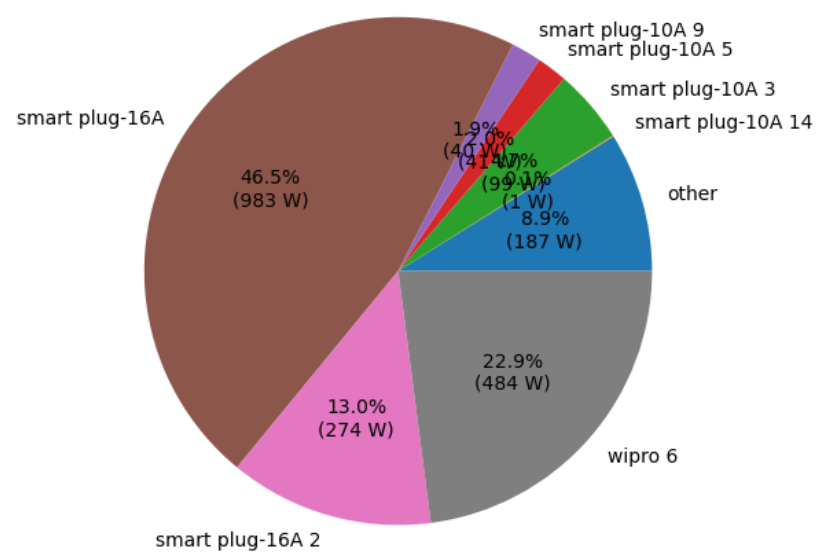


**Appliance-Type-Wise Distribution**



**Appliance Energy Disaggregation – Category /Type/ Name**



While the results in this case are accurate, this is not the case every time. The classification algorithm used here has not performed as well. The estimation algorithm is also an approximation based on the mean consumption by the appliance and hence not accurate enough.

**Steps to Work on**

1. **Appliance\_Classification()**
2. **Appliance\_Energy\_Estimation()**
3. **Fine tuning the Event Detection Algorithm**