In [2]:
```
%matplotlib inline
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import json
from pandas.io.json import json_normalize #package for flattening json in pandas df
import matplotlib.pyplot as plt
from datetime import date, timedelta, datetime
```

In [3]:
```
import os
os.getcwd()
```

Out[3]:
```
'C:\\Users\\Rockwell\\Documents\\GitHub\\Demand-Response'
```

1. Deal with load data

We can find that TOU has complete data in 2013

In [3]:
```
df_tou = pd.read_csv("C:\\Users\\Rockwell\\Desktop\\Paper4\\data_collection\\data_tables\\consumption_d.csv")
df_tou
```

Out[3]:

	GMT	D0000	D0001	D0002	D0003	D0004	D0005	D0006	D0007	D0008	...	D1015	D1016	D1017	D1018	D1019	D1020	D
0	2011-11-23 09:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
1	2011-11-23 09:30:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
2	2011-11-23 10:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
3	2011-11-23 10:30:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
4	2011-11-23 11:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
5	2011-11-23 11:30:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
6	2011-11-23 12:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
7	2011-11-23 12:30:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
8	2011-11-23 13:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
9	2011-11-23 13:30:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
10	2011-11-23 14:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
11	2011-11-23 14:30:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
12	2011-11-23 15:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
13	2011-11-23 15:30:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
14	2011-11-23 16:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
15	2011-11-23 16:30:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
16	2011-11-23 17:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
17	2011-11-23 17:30:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
18	2011-11-23 18:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
19	2011-11-23 18:30:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
20	2011-11-23 19:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N

	GMT	D0000	D0001	D0002	D0003	D0004	D0005	D0006	D0007	D0008	...	D1015	D1016	D1017	D1018	D1019	D1020	D
21	2011-11-23 19:30:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
22	2011-11-23 20:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
23	2011-11-23 20:30:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
24	2011-11-23 21:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
25	2011-11-23 21:30:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
26	2011-11-23 22:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
27	2011-11-23 22:30:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
28	2011-11-23 23:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
29	2011-11-23 23:30:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
39697	2014-02-27 09:30:00	0.060	0.134	0.139	0.031	0.068	0.118	0.178	0.151	0.041	...	0.139	0.014	0.229	0.049	0.199	0.099	0.
39698	2014-02-27 10:00:00	0.034	0.117	0.135	0.005	0.295	0.153	0.128	0.101	0.068	...	0.151	0.013	0.204	0.055	0.441	0.161	0.
39699	2014-02-27 10:30:00	0.069	0.154	0.115	0.042	0.370	0.134	0.105	0.084	0.071	...	0.034	0.033	0.126	0.071	0.186	0.151	0.
39700	2014-02-27 11:00:00	0.035	0.034	0.327	0.141	0.431	0.176	0.154	0.283	0.043	...	0.054	0.132	0.149	0.046	0.160	0.106	0.
39701	2014-02-27 11:30:00	0.056	0.104	0.467	0.089	0.048	0.252	0.146	0.200	0.120	...	0.041	0.013	0.290	0.059	0.136	0.100	0.
39702	2014-02-27 12:00:00	0.262	0.040	0.489	0.060	0.063	0.123	0.127	0.216	0.082	...	0.020	0.006	0.135	0.070	0.082	0.174	0.
39703	2014-02-27 12:30:00	0.114	0.035	0.136	0.223	0.065	0.149	0.154	0.296	0.048	...	0.028	0.006	0.101	0.038	0.163	0.185	0.
39704	2014-02-27 13:00:00	0.147	0.041	0.117	0.053	0.176	0.125	0.191	2.246	0.081	...	0.060	0.014	0.124	0.066	0.054	0.160	0.
39705	2014-02-27 13:30:00	0.034	0.030	0.170	0.052	0.150	0.115	0.059	0.231	0.080	...	0.017	0.046	0.188	0.063	0.094	0.165	0.
39706	2014-02-27 14:00:00	0.174	0.041	0.151	0.051	0.078	0.164	0.019	0.202	0.125	...	0.039	0.157	0.145	0.039	0.089	0.121	0.
39707	2014-02-27 14:30:00	0.136	0.033	0.250	0.052	0.117	0.161	0.029	0.133	0.123	...	0.016	0.014	0.162	0.072	0.091	0.121	0.
39708	2014-02-27 15:00:00	0.151	0.062	0.220	0.051	0.162	0.180	0.268	0.114	0.094	...	0.104	0.013	0.143	0.048	0.100	0.132	0.

	GMT	D0000	D0001	D0002	D0003	D0004	D0005	D0006	D0007	D0008	...	D1015	D1016	D1017	D1018	D1019	D1020	D
39709	2014-02-27 15:30:00	0.131	0.055	0.115	0.194	0.076	0.154	0.320	0.188	0.148	...	0.048	0.015	0.367	0.056	0.090	0.136	0.
39710	2014-02-27 16:00:00	0.283	0.050	0.110	0.458	0.148	0.136	0.512	0.133	0.189	...	0.110	0.034	0.254	0.067	0.093	0.108	0.
39711	2014-02-27 16:30:00	0.221	0.097	1.391	0.228	0.169	0.154	0.497	0.139	0.094	...	0.113	0.117	0.143	0.035	0.090	0.045	0.
39712	2014-02-27 17:00:00	0.167	0.120	0.673	0.086	0.239	0.154	0.755	0.116	0.134	...	0.096	0.014	0.155	0.074	0.090	0.063	0.
39713	2014-02-27 17:30:00	0.171	0.094	1.171	0.086	0.237	0.219	0.423	0.141	0.113	...	0.187	0.013	0.134	0.044	0.175	0.481	0.
39714	2014-02-27 18:00:00	0.243	0.098	1.959	0.093	0.350	0.241	0.499	0.228	0.162	...	0.095	0.020	0.150	0.066	0.162	1.010	0.
39715	2014-02-27 18:30:00	0.290	0.083	0.845	0.086	0.369	0.190	0.422	0.148	0.277	...	0.172	0.001	0.199	0.107	0.337	0.827	0.
39716	2014-02-27 19:00:00	0.305	0.058	1.247	0.048	0.281	0.215	0.868	0.199	0.316	...	0.174	0.034	0.165	0.076	0.252	0.657	0.
39717	2014-02-27 19:30:00	0.307	0.066	2.084	0.058	0.361	0.238	1.265	0.178	0.338	...	0.207	0.019	0.157	0.079	0.259	0.258	0.
39718	2014-02-27 20:00:00	0.322	0.061	1.511	0.100	0.271	0.186	0.749	0.136	0.815	...	0.266	0.020	0.136	0.055	0.224	0.282	0.
39719	2014-02-27 20:30:00	0.173	0.042	0.610	0.097	0.240	0.226	0.258	0.160	0.337	...	0.067	0.135	0.138	0.053	0.201	0.278	0.
39720	2014-02-27 21:00:00	0.212	0.036	0.536	0.096	0.200	0.239	0.287	0.185	0.561	...	0.116	0.020	0.159	0.096	0.274	0.263	0.
39721	2014-02-27 21:30:00	0.178	0.037	0.404	0.096	0.218	0.186	0.458	0.177	0.916	...	0.073	0.043	0.222	0.074	0.390	0.270	0.
39722	2014-02-27 22:00:00	0.401	0.046	0.457	0.095	0.203	0.232	0.284	0.126	0.528	...	0.089	0.031	0.139	0.085	0.393	0.324	0.
39723	2014-02-27 22:30:00	0.190	0.184	0.430	0.096	0.176	0.240	0.310	0.126	0.650	...	0.033	0.020	0.132	0.085	0.388	0.268	0.
39724	2014-02-27 23:00:00	0.119	0.108	0.309	0.087	0.098	0.195	0.236	0.161	0.182	...	0.048	0.021	0.104	0.037	0.312	0.244	0.
39725	2014-02-27 23:30:00	0.042	0.097	0.353	0.052	0.020	0.225	0.344	0.164	0.077	...	0.037	0.020	0.093	0.067	0.208	0.296	0.
39726	2014-02-28 00:00:00	0.056	0.070	0.289	0.052	0.017	0.304	0.331	0.125	0.066	...	0.019	0.020	0.138	0.273	0.120	0.263	0.

39727 rows × 1026 columns

We need to extract all the TOU load data in 2013

In [4]:
```
df_tou_2013 = df_tou.loc[df_tou['GMT'].str[0:4] == '2013'].reset_index(drop = True)
# Store the complete dataframe to a csv file
df_tou_2013.to_csv("C:\\Users\\Rockwell\\Desktop\\Paper4\\data_collection\\data_tables\\Consumption_tou2013.csv", index = False)
df_tou_2013
```

Out[4]:

	GMT	D0000	D0001	D0002	D0003	D0004	D0005	D0006	D0007	D0008	...	D1015	D1016	D1017	D1018	D1019	D1020	D
0	2013-01-01 00:00:00	1.043	0.181	0.224	0.077	0.195	0.040	0.146	0.131	0.214	...	0.016	0.028	0.266	0.063	0.182	0.305	0.
1	2013-01-01 00:30:00	0.404	0.248	0.227	0.078	0.202	0.066	0.161	0.137	0.176	...	0.032	0.005	0.236	0.054	0.196	0.838	0.
2	2013-01-01 01:00:00	0.185	0.206	0.231	0.076	0.188	0.057	0.227	0.164	0.187	...	0.046	0.005	0.289	0.060	0.218	0.446	0.
3	2013-01-01 01:30:00	0.151	0.171	0.236	0.077	0.039	0.052	0.138	0.136	0.341	...	0.016	0.005	0.239	0.057	0.181	0.410	0.
4	2013-01-01 02:00:00	0.139	0.068	0.239	0.077	0.024	0.070	0.040	0.103	0.087	...	0.016	0.005	0.120	0.263	0.174	0.282	0.
5	2013-01-01 02:30:00	0.105	0.083	0.251	0.077	0.102	0.038	0.020	0.094	0.063	...	0.016	0.005	0.097	0.047	0.188	0.239	0.
6	2013-01-01 03:00:00	0.122	0.083	0.246	0.077	0.033	0.083	0.035	0.086	0.069	...	0.046	0.005	0.117	0.065	0.194	0.275	0.
7	2013-01-01 03:30:00	0.091	0.072	0.334	0.076	0.024	0.026	0.032	0.081	0.082	...	0.016	0.006	0.089	0.054	0.174	0.195	0.
8	2013-01-01 04:00:00	0.110	0.060	0.287	0.079	0.033	0.083	0.020	0.073	0.054	...	0.015	0.051	0.104	0.056	0.164	0.106	0.
9	2013-01-01 04:30:00	0.100	0.061	0.292	0.077	0.028	0.024	0.036	0.071	0.079	...	0.046	0.045	0.378	0.062	0.190	0.140	0.
10	2013-01-01 05:00:00	0.091	0.060	0.260	0.077	0.027	0.102	0.016	0.072	0.074	...	0.016	0.006	0.234	0.048	0.185	0.087	0.
11	2013-01-01 05:30:00	0.107	0.068	0.249	0.871	0.049	0.092	0.036	0.071	0.053	...	0.044	0.005	0.175	0.070	0.176	0.146	0.
12	2013-01-01 06:00:00	0.084	0.076	0.278	1.644	0.096	0.123	0.020	0.077	0.080	...	0.016	0.005	0.168	0.068	0.175	0.095	0.
13	2013-01-01 06:30:00	0.118	0.044	0.316	1.198	0.062	0.048	0.038	0.100	0.072	...	0.033	0.005	0.138	0.114	0.185	0.099	0.
14	2013-01-01 07:00:00	0.086	0.063	0.312	0.457	0.031	0.097	0.064	0.239	0.054	...	0.015	0.005	0.087	0.051	0.170	0.130	0.
15	2013-01-01 07:30:00	0.081	0.063	0.304	0.077	0.031	0.073	0.033	0.178	0.074	...	0.046	0.006	0.084	0.068	0.173	0.092	0.
16	2013-01-01 08:00:00	0.054	0.082	0.272	0.076	0.034	0.098	0.034	0.109	0.078	...	0.015	0.005	0.086	0.052	0.127	0.105	0.
17	2013-01-01 08:30:00	0.306	0.241	0.253	0.077	0.022	0.092	0.019	0.068	0.097	...	0.032	0.220	0.136	0.060	0.081	0.124	0.
18	2013-01-01 09:00:00	0.105	0.220	0.244	0.077	0.033	0.052	0.079	0.061	0.076	...	0.016	0.065	0.117	0.084	0.077	0.079	0.
19	2013-01-01 09:30:00	0.073	0.243	0.238	0.076	0.198	0.084	0.017	0.116	0.092	...	0.046	0.041	0.128	0.167	0.088	0.135	0.
20	2013-01-01 10:00:00	0.105	0.170	0.251	0.360	0.496	0.038	0.116	0.071	0.214	...	0.015	0.115	0.097	0.058	0.062	0.121	0.

	GMT	D0000	D0001	D0002	D0003	D0004	D0005	D0006	D0007	D0008	...	D1015	D1016	D1017	D1018	D1019	D1020	D
21	2013-01-01 10:30:00	0.097	0.125	0.230	0.076	0.288	0.087	0.063	0.085	0.089	...	0.016	0.018	0.348	0.250	0.291	0.110	0.
22	2013-01-01 11:00:00	0.084	0.116	0.231	0.076	0.200	0.029	0.110	0.124	0.054	...	0.033	0.017	0.210	0.092	0.355	0.135	0.
23	2013-01-01 11:30:00	0.104	0.098	0.268	0.076	0.137	0.081	0.088	0.156	0.071	...	0.045	0.018	0.228	0.213	0.042	0.077	0.
24	2013-01-01 12:00:00	0.066	0.144	0.271	0.076	0.135	0.044	0.120	0.207	0.081	...	0.016	0.017	0.234	0.120	0.176	0.215	0.
25	2013-01-01 12:30:00	0.088	0.075	0.218	0.076	0.069	0.063	0.270	0.118	0.054	...	0.015	0.018	0.215	0.114	0.065	0.125	0.
26	2013-01-01 13:00:00	0.209	0.055	0.215	0.077	0.110	0.060	0.488	0.103	0.071	...	0.015	0.018	0.201	0.231	0.060	0.690	0.
27	2013-01-01 13:30:00	0.220	0.085	0.224	0.075	0.168	0.153	1.359	0.127	0.083	...	0.045	0.049	0.207	0.482	0.073	0.223	0.
28	2013-01-01 14:00:00	0.248	0.068	0.217	0.291	0.157	0.181	0.288	0.131	0.354	...	0.015	0.063	0.366	0.427	0.073	0.261	0.
29	2013-01-01 14:30:00	0.246	0.083	0.223	0.130	0.130	0.179	0.232	0.174	0.130	...	0.015	0.029	0.811	0.153	0.075	0.583	0.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
17490	2013-12-31 09:00:00	0.070	0.247	0.153	0.051	0.198	0.103	0.167	0.146	0.279	...	0.049	0.037	0.082	0.049	0.032	0.115	0.
17491	2013-12-31 09:30:00	0.037	0.253	0.125	0.050	0.137	0.196	0.307	0.121	0.357	...	0.032	0.019	0.128	0.068	0.029	0.072	0.
17492	2013-12-31 10:00:00	0.052	0.183	0.150	0.049	0.203	0.134	0.293	0.129	0.140	...	0.028	0.016	0.122	0.061	0.029	0.110	0.
17493	2013-12-31 10:30:00	0.054	0.130	0.156	0.049	0.149	0.120	0.124	0.610	0.041	...	0.025	0.017	0.118	0.043	0.168	0.122	0.
17494	2013-12-31 11:00:00	0.037	0.076	0.123	0.049	0.172	0.346	0.226	0.486	0.074	...	0.048	0.017	0.282	0.074	0.132	0.122	0.
17495	2013-12-31 11:30:00	0.072	0.031	0.149	0.050	0.096	0.075	1.124	0.318	0.055	...	0.018	0.157	0.120	0.036	0.164	0.072	0.
17496	2013-12-31 12:00:00	0.037	0.032	0.147	0.049	0.128	0.078	0.815	0.196	0.054	...	0.017	0.016	0.108	0.069	0.081	0.091	0.
17497	2013-12-31 12:30:00	0.049	0.130	0.170	0.049	0.182	0.165	2.150	0.203	0.071	...	0.034	0.017	0.158	0.051	0.075	0.136	0.
17498	2013-12-31 13:00:00	0.057	0.091	0.206	0.049	0.237	0.138	0.686	0.203	0.187	...	0.035	0.017	0.624	0.053	0.067	0.233	0.
17499	2013-12-31 13:30:00	0.037	0.113	0.203	0.049	0.262	0.180	0.771	0.215	0.096	...	0.052	0.018	0.228	0.065	0.067	0.504	0.
17500	2013-12-31 14:00:00	0.077	0.106	0.150	0.224	0.233	0.177	0.543	0.190	0.089	...	0.017	0.017	0.225	0.038	0.075	0.266	0.
17501	2013-12-31 14:30:00	0.046	0.084	0.155	0.050	0.145	0.178	1.294	0.168	0.144	...	0.018	0.142	0.177	0.075	0.199	0.354	0.

	GMT	D0000	D0001	D0002	D0003	D0004	D0005	D0006	D0007	D0008	...	D1015	D1016	D1017	D1018	D1019	D1020	D
17502	2013-12-31 15:00:00	0.200	0.162	0.160	0.468	0.139	0.065	1.595	0.186	0.102	...	0.060	0.017	0.170	0.037	0.118	0.275	0.
17503	2013-12-31 15:30:00	0.356	0.223	0.135	0.339	0.051	0.138	0.745	0.194	0.093	...	0.019	0.022	0.266	0.070	0.079	0.161	0.
17504	2013-12-31 16:00:00	0.181	0.123	0.171	0.053	0.006	0.137	0.720	0.188	0.188	...	0.018	0.188	0.140	0.046	0.090	0.084	0.
17505	2013-12-31 16:30:00	0.202	0.067	0.157	0.061	0.012	0.168	0.651	0.187	0.152	...	0.037	0.024	0.174	0.060	0.146	0.125	0.
17506	2013-12-31 17:00:00	0.192	0.061	0.154	0.091	0.022	0.225	0.746	0.193	0.196	...	0.048	0.036	0.166	0.303	0.060	0.118	0.
17507	2013-12-31 17:30:00	0.507	0.168	0.158	0.094	0.029	0.202	0.616	0.204	0.212	...	0.041	0.042	0.161	0.304	0.052	0.098	0.
17508	2013-12-31 18:00:00	0.355	0.151	0.164	0.093	0.020	0.228	0.668	0.268	0.220	...	0.075	0.116	0.169	0.455	0.026	0.070	0.
17509	2013-12-31 18:30:00	0.373	0.216	0.159	0.169	0.015	0.298	0.839	0.280	0.274	...	0.060	0.001	0.212	0.125	0.020	0.131	0.
17510	2013-12-31 19:00:00	0.247	0.265	0.156	0.092	0.006	0.225	0.932	0.251	0.293	...	0.087	0.000	0.258	0.077	0.338	0.118	0.
17511	2013-12-31 19:30:00	0.323	0.165	0.173	0.093	0.020	0.212	0.744	0.232	0.450	...	0.077	0.009	0.160	0.044	0.948	0.119	0.
17512	2013-12-31 20:00:00	0.346	0.179	0.164	0.094	0.016	0.263	0.477	0.252	0.233	...	0.050	0.054	0.136	0.062	0.471	0.146	0.
17513	2013-12-31 20:30:00	0.238	0.163	0.154	0.093	0.022	0.259	0.358	0.282	0.419	...	0.072	0.024	0.248	0.050	0.364	0.198	0.
17514	2013-12-31 21:00:00	0.335	0.145	0.163	0.093	0.011	0.223	0.325	0.255	0.222	...	0.070	0.131	0.260	0.054	0.449	0.183	0.
17515	2013-12-31 21:30:00	0.144	0.227	0.176	0.093	0.026	0.220	0.292	0.527	0.185	...	0.028	0.024	0.251	0.058	0.408	0.236	0.
17516	2013-12-31 22:00:00	0.234	0.256	0.145	0.094	0.006	0.248	0.387	0.203	0.222	...	0.020	0.024	0.268	0.047	0.381	0.284	0.
17517	2013-12-31 22:30:00	0.182	0.117	0.177	0.090	0.006	0.232	0.327	0.174	0.182	...	0.044	0.025	0.351	0.064	0.361	0.662	0.
17518	2013-12-31 23:00:00	0.153	0.078	0.158	0.055	0.025	0.212	0.164	0.187	0.219	...	0.036	0.201	0.249	0.041	0.267	0.640	0.
17519	2013-12-31 23:30:00	0.166	0.025	0.162	0.049	0.012	0.241	0.132	0.195	0.196	...	0.037	0.024	0.246	0.069	0.291	0.295	0.

17520 rows × 1026 columns

In [5]:
```
# Now we compress the rows such that the frequency is 1 hour
# first let's delete the GMT index of matrix consists in odd rows
odd_df = df_tou_2013.loc[1::2].copy()
odd_df.loc[:, 'GMT'] = ''
# create the new dataframe that is the sum of odd and even rows
df_tou_2013_1h = (df_tou_2013[::2] + odd_df.values).reset_index(drop = True)
# store it to a csv file
df_tou_2013_1h.to_csv("C:\\Users\\Rockwell\\Desktop\\Paper4\\data_collection\\data_tables\\Consumption_tou201
3_1h.csv", index = False)
```

We can find that non-TOU has complete data in 2013

In [6]:
```
df_ntou = pd.read_csv("C:\\Users\\Rockwell\\Desktop\\Paper4\\data_collection\\data_tables\\consumption_n.csv"
)
df_ntou
```

Out[6]:

	GMT	N0000	N0001	N0002	N0003	N0004	N0005	N0006	N0007	N0008	...	N4163	N4164	N4165	N4166	N4167	N4168	N
0	2011-11-23 09:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
1	2011-11-23 09:30:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
2	2011-11-23 10:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
3	2011-11-23 10:30:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
4	2011-11-23 11:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
5	2011-11-23 11:30:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
6	2011-11-23 12:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
7	2011-11-23 12:30:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
8	2011-11-23 13:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
9	2011-11-23 13:30:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
10	2011-11-23 14:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
11	2011-11-23 14:30:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
12	2011-11-23 15:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
13	2011-11-23 15:30:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
14	2011-11-23 16:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
15	2011-11-23 16:30:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
16	2011-11-23 17:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
17	2011-11-23 17:30:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
18	2011-11-23 18:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
19	2011-11-23 18:30:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
20	2011-11-23 19:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N

	GMT	N0000	N0001	N0002	N0003	N0004	N0005	N0006	N0007	N0008	...	N4163	N4164	N4165	N4166	N4167	N4168	N
21	2011-11-23 19:30:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
22	2011-11-23 20:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
23	2011-11-23 20:30:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
24	2011-11-23 21:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
25	2011-11-23 21:30:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
26	2011-11-23 22:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
27	2011-11-23 22:30:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
28	2011-11-23 23:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
29	2011-11-23 23:30:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
39697	2014-02-27 09:30:00	0.165	NaN	0.136	0.151	0.125	0.118	0.046	0.174	0.221	...	0.089	0.197	0.058	0.021	0.174	0.153	N
39698	2014-02-27 10:00:00	0.188	NaN	0.096	0.123	0.180	0.112	0.150	0.184	0.142	...	0.086	0.048	0.041	0.010	0.335	0.115	N
39699	2014-02-27 10:30:00	0.221	NaN	0.705	0.089	0.175	0.566	0.082	0.115	0.144	...	0.187	0.014	0.045	0.010	0.194	0.084	N
39700	2014-02-27 11:00:00	0.188	NaN	0.541	0.091	0.789	0.217	0.117	0.096	0.207	...	0.083	0.040	0.057	0.010	1.095	0.160	N
39701	2014-02-27 11:30:00	0.176	NaN	0.279	0.716	0.354	0.073	0.127	0.098	0.152	...	0.111	0.026	0.046	0.028	1.364	0.027	N
39702	2014-02-27 12:00:00	0.664	NaN	0.201	0.172	0.171	0.193	0.096	0.081	0.181	...	0.091	0.075	0.039	0.039	1.687	0.045	N
39703	2014-02-27 12:30:00	0.282	NaN	0.067	0.172	0.351	0.148	0.070	0.091	0.139	...	0.099	0.029	0.056	0.427	1.065	0.032	N
39704	2014-02-27 13:00:00	0.143	NaN	0.265	0.526	0.204	0.232	0.050	0.088	0.081	...	0.092	0.075	0.053	0.487	0.688	0.028	N
39705	2014-02-27 13:30:00	0.080	NaN	0.050	1.598	0.452	0.165	0.036	0.069	0.068	...	0.103	0.026	0.033	0.041	0.509	0.028	N
39706	2014-02-27 14:00:00	0.111	NaN	0.065	0.552	0.367	0.108	0.040	0.089	0.053	...	0.128	0.070	0.058	0.010	0.346	0.046	N
39707	2014-02-27 14:30:00	0.107	NaN	0.122	0.237	0.334	0.119	0.035	0.081	0.102	...	0.114	0.057	0.057	0.429	0.112	0.028	N
39708	2014-02-27 15:00:00	0.120	NaN	0.242	0.214	0.310	0.260	0.025	0.073	0.112	...	0.071	0.071	0.030	0.162	0.392	0.028	N

	GMT	N0000	N0001	N0002	N0003	N0004	N0005	N0006	N0007	N0008	...	N4163	N4164	N4165	N4166	N4167	N4168	N
39709	2014-02-27 15:30:00	1.042	NaN	0.213	0.197	0.231	0.062	0.052	0.339	0.200	...	0.070	0.106	0.062	0.021	0.519	0.041	N
39710	2014-02-27 16:00:00	1.106	NaN	0.278	0.187	0.194	0.034	0.050	0.600	0.089	...	0.157	0.092	0.078	0.021	0.204	0.034	N
39711	2014-02-27 16:30:00	0.151	NaN	0.167	0.134	0.050	0.277	0.062	0.087	0.068	...	0.188	0.141	0.047	0.092	0.233	0.058	N
39712	2014-02-27 17:00:00	0.140	NaN	0.073	0.164	0.076	0.144	0.044	0.160	0.085	...	0.134	0.259	0.065	0.875	0.168	0.059	N
39713	2014-02-27 17:30:00	0.164	NaN	0.752	0.359	0.092	0.477	0.027	0.102	0.094	...	0.163	0.103	0.062	0.878	0.171	0.074	N
39714	2014-02-27 18:00:00	0.183	NaN	0.146	0.218	0.082	0.933	0.050	0.101	0.071	...	0.127	0.102	0.035	0.978	0.214	0.076	N
39715	2014-02-27 18:30:00	0.157	NaN	0.187	0.215	0.077	0.534	0.038	0.245	0.106	...	0.151	0.115	0.112	0.824	0.561	0.154	N
39716	2014-02-27 19:00:00	0.205	NaN	0.375	0.263	0.078	0.306	0.063	0.200	0.103	...	0.095	0.102	0.144	0.875	0.689	0.202	N
39717	2014-02-27 19:30:00	0.287	NaN	0.174	0.513	0.117	0.505	0.035	0.197	0.072	...	0.200	0.143	0.092	0.858	0.769	0.298	N
39718	2014-02-27 20:00:00	0.266	NaN	0.170	0.362	0.505	0.409	0.067	0.198	0.089	...	0.151	0.115	0.104	0.702	0.817	0.195	N
39719	2014-02-27 20:30:00	0.390	NaN	0.192	0.422	0.552	0.316	0.066	0.194	0.085	...	0.068	0.098	0.120	0.718	0.306	0.143	N
39720	2014-02-27 21:00:00	0.331	NaN	0.112	0.758	1.118	0.363	0.155	0.201	0.064	...	0.063	0.114	0.094	0.711	1.435	0.132	N
39721	2014-02-27 21:30:00	0.262	NaN	0.053	1.147	0.720	0.322	0.040	0.174	0.074	...	0.074	0.064	0.099	0.189	0.302	0.128	N
39722	2014-02-27 22:00:00	0.250	NaN	0.020	0.471	0.422	0.399	0.052	0.102	0.103	...	0.819	0.072	0.110	0.192	0.452	0.122	N
39723	2014-02-27 22:30:00	0.141	NaN	0.020	0.150	0.508	0.231	0.079	0.085	0.101	...	0.275	0.015	0.096	0.009	0.294	0.140	N
39724	2014-02-27 23:00:00	0.083	NaN	0.043	0.119	0.225	0.016	0.088	0.087	0.121	...	0.233	0.022	0.096	0.319	0.226	0.128	N
39725	2014-02-27 23:30:00	0.117	NaN	0.089	0.110	0.215	0.061	0.024	0.067	0.459	...	0.177	0.042	0.081	0.040	0.203	0.136	N
39726	2014-02-28 00:00:00	0.115	NaN	0.085	0.094	0.165	0.035	0.024	0.089	0.164	...	0.148	0.007	0.048	0.034	0.161	0.101	N

39727 rows × 4174 columns

We then extract all the non-TOU load data in 2013

In [7]:
```
df_ntou_2013 = df_ntou.loc[df_ntou['GMT'].str[0:4] == '2013'].reset_index(drop = True)
df_ntou_2013.to_csv("C:\\Users\\Rockwell\\Desktop\\Paper4\\data_collection\\data_tables\\Consumption_Ntou201
3.csv", index = False)
df_ntou_2013
```

Out[7]:

	GMT	N0000	N0001	N0002	N0003	N0004	N0005	N0006	N0007	N0008	...	N4163	N4164	N4165	N4166	N4167	N4168	N
0	2013-01-01 00:00:00	0.306	0.117	0.018	0.140	0.090	0.369	0.000	0.077	0.170	...	0.125	0.003	0.044	0.010	0.338	0.099	0.
1	2013-01-01 00:30:00	0.732	0.115	0.034	0.116	0.091	0.404	0.000	0.086	0.091	...	0.062	0.048	0.043	0.010	0.248	0.102	0.
2	2013-01-01 01:00:00	0.250	0.152	0.051	0.092	0.092	0.245	0.000	0.082	0.047	...	0.132	0.014	0.042	0.020	0.263	0.093	0.
3	2013-01-01 01:30:00	0.234	0.115	0.050	0.100	0.093	0.114	0.000	0.070	0.043	...	0.172	0.031	0.024	0.095	0.201	0.101	0.
4	2013-01-01 02:00:00	0.214	0.114	0.048	0.100	0.108	0.059	0.000	0.085	0.096	...	0.214	0.011	0.042	0.215	0.072	0.114	0.
5	2013-01-01 02:30:00	0.235	0.152	0.044	0.100	0.376	0.091	0.000	0.085	0.060	...	0.182	0.041	0.042	0.020	0.118	0.127	0.
6	2013-01-01 03:00:00	0.182	0.116	0.017	0.133	0.059	0.077	0.000	0.072	0.039	...	0.181	0.056	0.028	0.010	0.095	0.097	0.
7	2013-01-01 03:30:00	0.208	0.114	0.017	0.129	0.073	0.078	0.000	0.080	0.055	...	0.174	0.055	0.038	0.011	0.062	0.088	0.
8	2013-01-01 04:00:00	0.318	0.153	0.016	0.120	0.092	0.091	0.000	0.086	0.075	...	0.167	0.097	0.042	0.010	0.060	0.218	0.
9	2013-01-01 04:30:00	0.618	0.115	0.036	0.085	0.091	0.072	0.000	0.079	0.080	...	0.179	0.022	0.030	0.011	0.115	0.094	0.
10	2013-01-01 05:00:00	0.181	0.115	0.051	0.099	0.056	0.081	0.000	0.073	0.047	...	0.165	0.044	0.035	0.028	0.104	0.091	0.
11	2013-01-01 05:30:00	0.207	0.150	0.049	0.100	0.056	0.084	0.000	0.086	0.038	...	0.178	0.005	0.040	0.804	0.122	0.089	0.
12	2013-01-01 06:00:00	0.207	0.113	0.048	0.090	0.057	0.070	0.000	0.087	0.078	...	0.177	0.028	0.031	0.930	0.061	0.100	0.
13	2013-01-01 06:30:00	0.177	0.114	0.020	0.093	0.058	0.091	0.000	0.067	0.079	...	0.178	0.021	0.034	1.906	1.460	0.087	0.
14	2013-01-01 07:00:00	0.230	0.151	0.017	0.124	0.058	0.069	0.000	0.084	0.078	...	0.187	0.013	0.041	1.908	1.349	0.144	0.
15	2013-01-01 07:30:00	0.176	0.113	0.016	0.129	0.059	0.050	0.000	0.085	0.107	...	0.137	0.250	0.030	1.912	0.309	0.093	0.
16	2013-01-01 08:00:00	0.244	0.115	0.418	0.129	0.059	0.038	0.000	0.076	0.088	...	0.252	0.075	0.040	1.915	0.120	0.087	0.
17	2013-01-01 08:30:00	0.706	0.149	0.159	0.319	0.059	0.028	0.000	0.076	0.125	...	0.211	0.083	0.040	0.831	0.501	0.079	0.
18	2013-01-01 09:00:00	0.204	0.112	0.159	0.301	0.253	0.047	0.000	0.084	0.109	...	0.180	0.041	0.031	1.132	0.552	0.223	0.
19	2013-01-01 09:30:00	0.244	0.116	0.079	0.297	0.052	0.035	0.000	0.083	0.042	...	0.157	0.072	0.039	0.215	0.356	0.089	0.
20	2013-01-01 10:00:00	0.240	0.152	0.572	0.667	0.055	0.038	0.000	0.068	0.315	...	0.130	0.022	0.043	0.021	1.047	0.075	0.

	GMT	N0000	N0001	N0002	N0003	N0004	N0005	N0006	N0007	N0008	...	N4163	N4164	N4165	N4166	N4167	N4168	N
21	2013-01-01 10:30:00	0.154	0.114	0.278	1.827	0.061	0.066	0.000	0.084	0.233	...	0.152	0.060	0.059	0.011	0.515	0.063	0.
22	2013-01-01 11:00:00	0.257	0.113	0.140	0.832	0.079	0.198	0.000	0.084	0.114	...	0.208	0.056	0.027	0.011	1.428	0.073	0.
23	2013-01-01 11:30:00	0.221	0.152	0.222	0.682	0.190	0.323	0.000	0.072	0.087	...	0.177	0.055	0.054	0.010	0.349	0.035	0.
24	2013-01-01 12:00:00	0.244	0.116	0.061	0.163	0.117	0.305	0.000	0.078	0.107	...	0.272	0.068	0.062	0.011	0.328	0.168	0.
25	2013-01-01 12:30:00	0.219	0.115	0.081	0.210	0.199	0.494	0.000	0.085	0.105	...	0.551	0.058	0.051	0.010	1.653	0.769	0.
26	2013-01-01 13:00:00	0.342	0.152	0.134	0.202	0.142	1.498	0.000	0.078	0.178	...	0.187	0.093	0.034	0.039	1.792	0.140	0.
27	2013-01-01 13:30:00	0.305	0.115	0.078	0.286	0.874	1.897	0.000	0.070	0.262	...	0.195	0.056	0.045	0.046	1.207	0.070	0.
28	2013-01-01 14:00:00	0.240	0.115	0.050	0.252	0.974	1.680	0.000	0.085	0.076	...	0.151	0.072	0.044	0.011	0.809	0.375	0.
29	2013-01-01 14:30:00	0.214	0.152	0.117	0.240	0.619	0.836	0.000	0.085	0.138	...	0.198	0.050	0.030	0.010	0.374	0.119	0.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
17490	2013-12-31 09:00:00	0.236	0.223	0.064	0.241	0.370	0.098	0.036	0.133	0.155	...	0.041	0.041	0.057	0.249	0.525	0.201	N
17491	2013-12-31 09:30:00	0.179	0.414	0.195	0.267	0.485	0.022	0.149	0.139	0.147	...	0.038	0.024	0.059	0.134	1.430	0.105	N
17492	2013-12-31 10:00:00	0.194	0.415	0.264	0.240	0.294	0.022	0.163	0.105	0.150	...	0.044	0.057	0.038	0.018	1.680	0.225	N
17493	2013-12-31 10:30:00	0.165	0.267	0.377	0.280	0.297	0.066	0.151	0.125	0.103	...	0.033	0.010	0.051	0.018	0.585	0.071	N
17494	2013-12-31 11:00:00	0.225	0.246	0.176	0.324	0.368	0.013	0.081	0.117	0.096	...	0.047	0.044	0.057	0.025	0.664	0.075	N
17495	2013-12-31 11:30:00	0.166	0.342	0.154	0.257	0.141	0.034	0.088	0.097	0.093	...	0.026	0.013	0.054	0.029	1.143	0.031	N
17496	2013-12-31 12:00:00	0.218	0.264	0.153	0.689	0.087	0.056	0.174	0.108	0.093	...	0.053	0.015	0.033	0.859	0.959	0.052	N
17497	2013-12-31 12:30:00	0.182	0.187	0.369	0.702	0.067	0.081	0.187	0.092	0.103	...	0.018	0.043	0.057	0.324	0.563	0.064	N
17498	2013-12-31 13:00:00	0.154	0.197	0.798	0.479	0.226	0.253	0.081	0.101	0.140	...	0.088	0.066	0.060	0.166	0.953	0.035	N
17499	2013-12-31 13:30:00	0.176	0.302	0.341	0.237	1.217	0.164	0.129	0.109	0.133	...	0.024	0.166	0.062	0.075	1.094	0.041	N
17500	2013-12-31 14:00:00	0.157	0.301	0.060	0.213	1.039	0.136	0.073	0.090	0.130	...	0.050	0.098	0.040	0.018	1.168	0.070	N
17501	2013-12-31 14:30:00	0.150	0.226	0.059	0.205	0.606	0.164	0.095	0.596	0.128	...	0.055	0.098	0.086	0.018	0.233	0.032	N

	GMT	N0000	N0001	N0002	N0003	N0004	N0005	N0006	N0007	N0008	...	N4163	N4164	N4165	N4166	N4167	N4168	N
17502	2013-12-31 15:00:00	0.205	0.223	0.113	0.143	0.363	0.167	0.062	0.836	0.096	...	0.045	0.092	0.063	0.018	0.475	0.042	N
17503	2013-12-31 15:30:00	0.209	0.294	0.391	0.180	0.451	0.145	0.020	0.289	0.111	...	0.050	0.107	0.061	0.018	0.639	0.365	N
17504	2013-12-31 16:00:00	0.195	0.310	0.260	0.331	0.387	0.157	0.024	0.143	0.117	...	0.026	0.080	0.047	0.017	1.156	0.118	N
17505	2013-12-31 16:30:00	0.225	0.220	0.107	0.293	0.386	0.164	0.019	0.139	0.094	...	0.053	0.119	0.101	0.095	1.109	0.145	N
17506	2013-12-31 17:00:00	0.195	0.234	0.234	0.289	0.332	0.144	0.024	0.329	0.094	...	0.022	0.081	0.109	1.223	0.687	0.134	N
17507	2013-12-31 17:30:00	1.141	0.275	0.059	0.302	0.128	0.217	0.017	0.339	0.091	...	0.058	0.119	0.124	1.435	0.668	0.204	N
17508	2013-12-31 18:00:00	0.338	0.305	0.067	0.323	0.120	0.290	0.063	0.379	0.117	...	0.016	0.166	0.121	1.463	0.722	0.204	N
17509	2013-12-31 18:30:00	0.294	0.239	0.166	0.336	0.124	0.441	0.072	0.343	0.150	...	0.057	0.087	0.131	1.491	1.810	0.184	N
17510	2013-12-31 19:00:00	0.213	0.224	0.139	0.287	0.153	0.309	0.098	0.365	0.148	...	0.024	0.115	0.130	1.434	1.275	0.090	N
17511	2013-12-31 19:30:00	0.196	0.279	0.090	0.604	0.135	0.148	0.094	0.148	0.107	...	0.047	0.082	0.105	1.498	1.400	0.080	N
17512	2013-12-31 20:00:00	0.223	0.312	0.091	0.432	0.124	0.144	0.114	0.924	0.094	...	0.031	0.120	0.099	1.317	0.710	0.098	N
17513	2013-12-31 20:30:00	0.252	0.253	0.127	0.308	0.122	0.136	0.149	0.250	0.095	...	0.039	0.090	0.117	1.433	0.249	0.078	N
17514	2013-12-31 21:00:00	0.241	0.207	0.078	0.715	0.128	0.151	0.108	0.227	0.094	...	0.038	0.088	0.107	1.237	1.411	0.082	N
17515	2013-12-31 21:30:00	0.207	0.292	0.030	0.478	0.186	0.150	0.096	0.216	0.107	...	0.032	0.114	0.119	0.243	0.811	0.099	N
17516	2013-12-31 22:00:00	0.203	0.495	0.020	0.742	0.146	0.128	0.099	0.309	0.120	...	0.047	0.037	0.098	0.226	0.955	0.083	N
17517	2013-12-31 22:30:00	0.116	0.290	0.045	0.265	0.124	0.158	0.155	0.187	0.097	...	0.022	0.074	0.089	0.018	1.006	0.086	N
17518	2013-12-31 23:00:00	0.122	0.267	0.055	0.213	0.130	0.152	0.106	0.214	0.132	...	0.055	0.012	0.106	0.018	0.398	0.090	N
17519	2013-12-31 23:30:00	0.149	0.335	0.054	0.091	0.156	0.126	0.036	0.261	0.131	...	0.017	0.015	0.112	0.407	0.152	0.093	N

17520 rows × 4174 columns

In [8]:
```
# Now we compress the rows such that the frequency is 1 hour
# first let's delete the GMT index of matrix consists in odd rows
odd_df_ntou = df_ntou_2013.loc[1::2].copy()
odd_df_ntou.loc[:, 'GMT'] = ''
# create the new dataframe that is the sum of odd and even rows
df_ntou_2013_1h = (df_ntou_2013[::2] + odd_df_ntou.values).reset_index(drop = True)
# store it to a csv file
df_ntou_2013_1h.to_csv("C:\\Users\\Rockwell\\Desktop\\Paper4\\data_collection\\data_tables\\Consumption_Ntou2
013_1h.csv", index = False)
```

2. Write a function that can extract all key values of data, time and weather to a csv file from London weather 2013

1) read txt data by line

2) extract the date and time and temperature out of it

3) store them to a dataframe

4) save the dataframe as csv file "London_weather.csv"

5) combine data frame of all months together

In [9]:
```
with open("C:\\Users\\Rockwell\\Desktop\\Paper4\\data_collection\\weather\\LondonWeather2013_test.txt") as file_test:
    file_content = file_test.readlines()
```

In [10]:
```
weather_dict = {} # store the data in dictionary "time": "temperature"
for line in file_content:
    if ('"date":' in line) or ('"time":' in line) or ('"tempC":' in line) or ('"tempF":' in line):
        if '"date":' in line: # use the newest date
            newest_date = line.strip().strip('"date": ').strip(',').strip('"') # only keep date strings
        if '"time":' in line: # convert the time to form HH:MM:SS
            # clean the string
            cleaned_time = line.strip().strip('"time": ').strip(',').strip('"')
            if len(cleaned_time) == 1: # i.e. cleaned_time == '0'
                standard_time = '00:00:00'
            elif len(cleaned_time) == 3: # i.e. cleaned_time == '300' - '900' etc
                standard_time = '0' + cleaned_time[0] + ':' + '00:00'
            else:
                standard_time = cleaned_time[:2] + ':00:00'
            newest_key = newest_date + ' ' + standard_time
#             print(newest_key)
        if '"tempC":' in line: # store the tempC to the dictionary
            cleaned_tempC = line.strip().strip('"tempC": ').strip(',').strip('"')
        if '"tempF":' in line: # store the tempF to the dictionary
            cleaned_tempF = line.strip().strip('"tempF": ').strip(',').strip('"')
            weather_dict[newest_key] = [float(cleaned_tempC), float(cleaned_tempF)]
# print(weather_dict)

#convert the dictionary to dataframe
df_temp = pd.DataFrame.from_dict(weather_dict, orient = 'index',
                                 columns = ['TempC', 'TempF'])
df_temp = df_temp.reset_index()
df_temp.rename(columns={'index' : 'GMT'})
```

Out[10]:

	GMT	TempC	TempF
0	2013-01-01 00:00:00	7.0	45.0
1	2013-01-01 03:00:00	6.0	43.0
2	2013-01-01 06:00:00	4.0	38.0
3	2013-01-01 09:00:00	3.0	38.0
4	2013-01-01 12:00:00	6.0	43.0
5	2013-01-01 15:00:00	6.0	43.0
6	2013-01-01 18:00:00	4.0	39.0
7	2013-01-01 21:00:00	3.0	37.0
8	2013-01-02 00:00:00	2.0	36.0
9	2013-01-02 03:00:00	2.0	36.0
10	2013-01-02 06:00:00	2.0	35.0
11	2013-01-02 09:00:00	3.0	38.0
12	2013-01-02 12:00:00	5.0	41.0
13	2013-01-02 15:00:00	7.0	45.0
14	2013-01-02 18:00:00	9.0	48.0
15	2013-01-02 21:00:00	9.0	48.0
16	2013-01-03 00:00:00	8.0	46.0
17	2013-01-03 03:00:00	8.0	46.0
18	2013-01-03 06:00:00	9.0	48.0
19	2013-01-03 09:00:00	9.0	48.0
20	2013-01-03 12:00:00	12.0	53.0
21	2013-01-03 15:00:00	12.0	54.0
22	2013-01-03 18:00:00	10.0	50.0
23	2013-01-03 21:00:00	9.0	49.0
24	2013-01-04 00:00:00	7.0	45.0
25	2013-01-04 03:00:00	8.0	46.0
26	2013-01-04 06:00:00	8.0	47.0
27	2013-01-04 09:00:00	8.0	47.0
28	2013-01-04 12:00:00	11.0	51.0
29	2013-01-04 15:00:00	10.0	51.0
...	...	...	...
226	2013-01-29 06:00:00	7.0	45.0
227	2013-01-29 09:00:00	9.0	48.0
228	2013-01-29 12:00:00	12.0	53.0
229	2013-01-29 15:00:00	12.0	54.0
230	2013-01-29 18:00:00	12.0	53.0
231	2013-01-29 21:00:00	11.0	52.0
232	2013-01-30 00:00:00	12.0	53.0
233	2013-01-30 03:00:00	12.0	53.0
234	2013-01-30 06:00:00	11.0	51.0
235	2013-01-30 09:00:00	9.0	48.0
236	2013-01-30 12:00:00	10.0	50.0
237	2013-01-30 15:00:00	9.0	49.0
238	2013-01-30 18:00:00	7.0	44.0
239	2013-01-30 21:00:00	5.0	41.0
240	2013-01-31 00:00:00	6.0	44.0
241	2013-01-31 03:00:00	6.0	42.0
242	2013-01-31 06:00:00	5.0	41.0
243	2013-01-31 09:00:00	7.0	44.0
244	2013-01-31 12:00:00	8.0	47.0

	GMT	TempC	TempF
245	2013-01-31 15:00:00	9.0	49.0
246	2013-01-31 18:00:00	8.0	47.0
247	2013-01-31 21:00:00	7.0	45.0
248	2013-02-01 00:00:00	4.0	40.0
249	2013-02-01 03:00:00	4.0	38.0
250	2013-02-01 06:00:00	3.0	38.0
251	2013-02-01 09:00:00	5.0	40.0
252	2013-02-01 12:00:00	6.0	42.0
253	2013-02-01 15:00:00	6.0	43.0
254	2013-02-01 18:00:00	6.0	42.0
255	2013-02-01 21:00:00	5.0	40.0

256 rows × 3 columns

Now let's make it to a function such that input is the file path and the output is a dataframe.

In [11]:
```
def temp_text_read(path):
    with open(path) as file_test:
        file_content = file_test.readlines()
    weather_dict = {} # store the data in dictionary "time": "temperature"
    for line in file_content:
        if ('"date":' in line) or ('"time":' in line) or ('"tempC":' in line) or ('"tempF":' in line):
            if '"date":' in line: # use the newest date
                newest_date = line.strip().strip('"date": ').strip(',').strip('"') # only keep date strings
            if '"time":' in line: # convert the time to form HH:MM:SS
                # clean the string
                cleaned_time = line.strip().strip('"time": ').strip(',').strip('"')
                if len(cleaned_time) == 1: # i.e. cleaned_time == '0'
                    standard_time = '00:00:00'
                elif len(cleaned_time) == 3: # i.e. cleaned_time == '300' - '900' etc
                    standard_time = '0' + cleaned_time[0] + ':' + '00:00'
                else:
                    standard_time = cleaned_time[:2] + ':00:00'
                newest_key = newest_date + ' ' + standard_time
    #             print(newest_key)
            if '"tempC":' in line: # store the tempC to the dictionary
                cleaned_tempC = line.strip().strip('"tempC": ').strip(',').strip('"')
            if '"tempF":' in line: # store the tempF to the dictionary
                cleaned_tempF = line.strip().strip('"tempF": ').strip(',').strip('"')
                weather_dict[newest_key] = [float(cleaned_tempC), float(cleaned_tempF)]
    # print(weather_dict)

    #convert the dictionary to dataframe
    df_temp = pd.DataFrame.from_dict(weather_dict, orient = 'index',
                                     columns = ['TempC', 'TempF'])
    df_temp = df_temp.reset_index()
    df_temp = df_temp.rename(columns={'index' : 'GMT'})
    return df_temp # a dataframe of GMT time and the corresponding temperature (C and F)
```

In [12]:
```
file_path = "C:\\Users\\Rockwell\\Desktop\\Paper4\\data_collection\\weather\\LondonWeather2013_test.txt"
temp_text_read(file_path)
```

Out[12]:

	GMT	TempC	TempF
0	2013-01-01 00:00:00	7.0	45.0
1	2013-01-01 03:00:00	6.0	43.0
2	2013-01-01 06:00:00	4.0	38.0
3	2013-01-01 09:00:00	3.0	38.0
4	2013-01-01 12:00:00	6.0	43.0
5	2013-01-01 15:00:00	6.0	43.0
6	2013-01-01 18:00:00	4.0	39.0
7	2013-01-01 21:00:00	3.0	37.0
8	2013-01-02 00:00:00	2.0	36.0
9	2013-01-02 03:00:00	2.0	36.0
10	2013-01-02 06:00:00	2.0	35.0
11	2013-01-02 09:00:00	3.0	38.0
12	2013-01-02 12:00:00	5.0	41.0
13	2013-01-02 15:00:00	7.0	45.0
14	2013-01-02 18:00:00	9.0	48.0
15	2013-01-02 21:00:00	9.0	48.0
16	2013-01-03 00:00:00	8.0	46.0
17	2013-01-03 03:00:00	8.0	46.0
18	2013-01-03 06:00:00	9.0	48.0
19	2013-01-03 09:00:00	9.0	48.0
20	2013-01-03 12:00:00	12.0	53.0
21	2013-01-03 15:00:00	12.0	54.0
22	2013-01-03 18:00:00	10.0	50.0
23	2013-01-03 21:00:00	9.0	49.0
24	2013-01-04 00:00:00	7.0	45.0
25	2013-01-04 03:00:00	8.0	46.0
26	2013-01-04 06:00:00	8.0	47.0
27	2013-01-04 09:00:00	8.0	47.0
28	2013-01-04 12:00:00	11.0	51.0
29	2013-01-04 15:00:00	10.0	51.0
...	...	...	...
226	2013-01-29 06:00:00	7.0	45.0
227	2013-01-29 09:00:00	9.0	48.0
228	2013-01-29 12:00:00	12.0	53.0
229	2013-01-29 15:00:00	12.0	54.0
230	2013-01-29 18:00:00	12.0	53.0
231	2013-01-29 21:00:00	11.0	52.0
232	2013-01-30 00:00:00	12.0	53.0
233	2013-01-30 03:00:00	12.0	53.0
234	2013-01-30 06:00:00	11.0	51.0
235	2013-01-30 09:00:00	9.0	48.0
236	2013-01-30 12:00:00	10.0	50.0
237	2013-01-30 15:00:00	9.0	49.0
238	2013-01-30 18:00:00	7.0	44.0
239	2013-01-30 21:00:00	5.0	41.0
240	2013-01-31 00:00:00	6.0	44.0
241	2013-01-31 03:00:00	6.0	42.0
242	2013-01-31 06:00:00	5.0	41.0
243	2013-01-31 09:00:00	7.0	44.0
244	2013-01-31 12:00:00	8.0	47.0

	GMT	TempC	TempF
245	2013-01-31 15:00:00	9.0	49.0
246	2013-01-31 18:00:00	8.0	47.0
247	2013-01-31 21:00:00	7.0	45.0
248	2013-02-01 00:00:00	4.0	40.0
249	2013-02-01 03:00:00	4.0	38.0
250	2013-02-01 06:00:00	3.0	38.0
251	2013-02-01 09:00:00	5.0	40.0
252	2013-02-01 12:00:00	6.0	42.0
253	2013-02-01 15:00:00	6.0	43.0
254	2013-02-01 18:00:00	6.0	42.0
255	2013-02-01 21:00:00	5.0	40.0

256 rows × 3 columns

Now process a batch of such files in 2013 and concatinate them together to store in a CSV file and a dataframe table.

In [13]:
```
pd.DataFrame(columns = ['GMT', 'TempC', 'TempF'])
```

Out[13]:

	GMT	TempC	TempF

In [14]:
```
# Create an empty dataframe
df = pd.DataFrame(columns = ['GMT', 'TempC', 'TempF'])

# Concatinate all dataframes that are read from json text files
for i in range(12):
    new_path = "C:\\Users\\Rockwell\\Desktop\\Paper4\\data_collection\\weather\\LondonWeather2013_" + str(i+1
) + '.txt'
    # Concatinate all and delete the repetitive rows
    df = pd.concat([df, temp_text_read(new_path)]).drop_duplicates().reset_index(drop = True)

# Store the complete dataframe to a csv file
df.to_csv("C:\\Users\\Rockwell\\Desktop\\Paper4\\data_collection\\weather\\LondonWeather2013_complete.csv", i
ndex = False)
```

We also want to interpolate data in each 3-hour interval, so that data is with 1-hour frequency. For this project, we use linear interpolation.

In [15]:
```
# First we construct a new dataframe that has GMT time column with 1-hour frequency.
weather_dict_extend = {}

start_date = date(2013, 1, 1)
end_date = date(2013,12,31)
# int(end_date - start_date)
date_span = int((end_date - start_date).days + 1)
for day in range(date_span):
    new_date = start_date + timedelta(day)
    day_str = new_date.strftime("%Y-%m-%d")
#     print(day_str)
    # repeat over all hours of a day
    for hour in range(24):
        if hour in range(10):
            hour_str = ' 0' + str(hour) + ':00:00'
        else:
            hour_str = ' ' + str(hour) + ':00:00'
        new_index = day_str + hour_str
        weather_dict_extend[new_index] = [None, None]

# we now get the extend dict and let's convert it to dataframe
# print(weather_dict_extend)
df_interp = pd.DataFrame.from_dict(weather_dict_extend, orient = 'index',
                                   columns = ['TempC', 'TempF'])
df_interp = df_interp.reset_index()
df_interp = df_interp.rename(columns={'index' : 'GMT'})
```

Having the extended empty dataframe, we can now do interpolation with the previous df.

In [16]:
```
for i in range(len(df) - 1):
    df_interp['TempC'][3 * i] = df['TempC'][i]
    df_interp['TempC'][3 * i + 1] = df['TempC'][i] + (df['TempC'][i + 1] - df['TempC'][i]) / 3
    df_interp['TempC'][3 * i + 2] = df['TempC'][i] + 2 * (df['TempC'][i + 1] - df['TempC'][i]) / 3

    df_interp['TempF'][3 * i] = df['TempF'][i]
    df_interp['TempF'][3 * i + 1] = df['TempF'][i] + (df['TempF'][i + 1] - df['TempF'][i]) / 3
    df_interp['TempF'][3 * i + 2] = df['TempF'][i] + 2 * (df['TempF'][i + 1] - df['TempF'][i]) / 3

#for the last interpolation, we just simply keep it as the same as the previous data
df_interp['TempC'][3 * (len(df) - 1)] = df['TempC'][len(df) - 1]
df_interp['TempC'][3 * (len(df) - 1) + 1] = df['TempC'][len(df) - 1]
df_interp['TempC'][3 * (len(df) - 1) + 2] = df['TempC'][len(df) - 1]

df_interp['TempF'][3 * (len(df) - 1)] = df['TempF'][len(df) - 1]
df_interp['TempF'][3 * (len(df) - 1) + 1] = df['TempF'][len(df) - 1]
df_interp['TempF'][3 * (len(df) - 1) + 2] = df['TempF'][len(df) - 1]

#store the interpolated data into a csv file
# Store the complete dataframe to a csv file
df_interp.to_csv("C:\\Users\\Rockwell\\Desktop\\Paper4\\data_collection\\weather\\LondonWeather2013_interpola
ted.csv", index = False)
df_interp
```

Out[16]:

	GMT	TempC	TempF
0	2013-01-01 00:00:00	7	45
1	2013-01-01 01:00:00	6.66667	44.3333
2	2013-01-01 02:00:00	6.33333	43.6667
3	2013-01-01 03:00:00	6	43
4	2013-01-01 04:00:00	5.33333	41.3333
5	2013-01-01 05:00:00	4.66667	39.6667
6	2013-01-01 06:00:00	4	38
7	2013-01-01 07:00:00	3.66667	38
8	2013-01-01 08:00:00	3.33333	38
9	2013-01-01 09:00:00	3	38
10	2013-01-01 10:00:00	4	39.6667
11	2013-01-01 11:00:00	5	41.3333
12	2013-01-01 12:00:00	6	43
13	2013-01-01 13:00:00	6	43
14	2013-01-01 14:00:00	6	43
15	2013-01-01 15:00:00	6	43
16	2013-01-01 16:00:00	5.33333	41.6667
17	2013-01-01 17:00:00	4.66667	40.3333
18	2013-01-01 18:00:00	4	39
19	2013-01-01 19:00:00	3.66667	38.3333
20	2013-01-01 20:00:00	3.33333	37.6667
21	2013-01-01 21:00:00	3	37
22	2013-01-01 22:00:00	2.66667	36.6667
23	2013-01-01 23:00:00	2.33333	36.3333
24	2013-01-02 00:00:00	2	36
25	2013-01-02 01:00:00	2	36
26	2013-01-02 02:00:00	2	36
27	2013-01-02 03:00:00	2	36
28	2013-01-02 04:00:00	2	35.6667
29	2013-01-02 05:00:00	2	35.3333
...	...	...	...
8730	2013-12-30 18:00:00	8	46
8731	2013-12-30 19:00:00	8	46
8732	2013-12-30 20:00:00	8	46
8733	2013-12-30 21:00:00	8	46
8734	2013-12-30 22:00:00	8.33333	46.6667
8735	2013-12-30 23:00:00	8.66667	47.3333
8736	2013-12-31 00:00:00	9	48
8737	2013-12-31 01:00:00	9	48
8738	2013-12-31 02:00:00	9	48
8739	2013-12-31 03:00:00	9	48
8740	2013-12-31 04:00:00	9	48
8741	2013-12-31 05:00:00	9	48
8742	2013-12-31 06:00:00	9	48
8743	2013-12-31 07:00:00	9	48
8744	2013-12-31 08:00:00	9	48
8745	2013-12-31 09:00:00	9	48
8746	2013-12-31 10:00:00	9	48
8747	2013-12-31 11:00:00	9	48
8748	2013-12-31 12:00:00	9	48

	GMT	TempC	TempF
8749	2013-12-31 13:00:00	9	48
8750	2013-12-31 14:00:00	9	48
8751	2013-12-31 15:00:00	9	48
8752	2013-12-31 16:00:00	9	48
8753	2013-12-31 17:00:00	9	48
8754	2013-12-31 18:00:00	9	48
8755	2013-12-31 19:00:00	9	48
8756	2013-12-31 20:00:00	9	48
8757	2013-12-31 21:00:00	9	48
8758	2013-12-31 22:00:00	9	48
8759	2013-12-31 23:00:00	9	48

8760 rows × 3 columns

3. With all the data available, we can visualize load and temperature over time in each month and observe the trend of these curves. (Maybe find some cycles of certain properties)

The data for TOU

C:\Users\Rockwell\Desktop\Paper4\data_collection\data_tables\Consumption_tou2013_1h.csv

The data for non-TOU

C:\Users\Rockwell\Desktop\Paper4\data_collection\data_tables\Consumption_Ntou2013_1h.csv

The data for weather

C:\Users\Rockwell\Desktop\Paper4\data_collection\weather\LondonWeather2013_interpolated.csv

In [3]:
```
df_tou1h = pd.read_csv("C:\\Users\\Rockwell\\Desktop\\Paper4\\data_collection\\data_tables\\Consumption_tou2013_1h.csv")
df_Ntou1h = pd.read_csv("C:\\Users\\Rockwell\\Desktop\\Paper4\\data_collection\\data_tables\\Consumption_Ntou2013_1h.csv")
df_wea1h = pd.read_csv("C:\\Users\\Rockwell\\Desktop\\Paper4\\data_collection\\weather\\LondonWeather2013_interpolated.csv")
```

In [3]:
```
print(df_tou1h.shape)
print(df_Ntou1h.shape)
print(df_wea1h.shape)
```

```
(8760, 1026)
(8760, 4174)
(8760, 3)
```

We can check the relationship between average energy consumption and outside temperature. (for the non-TOU case)

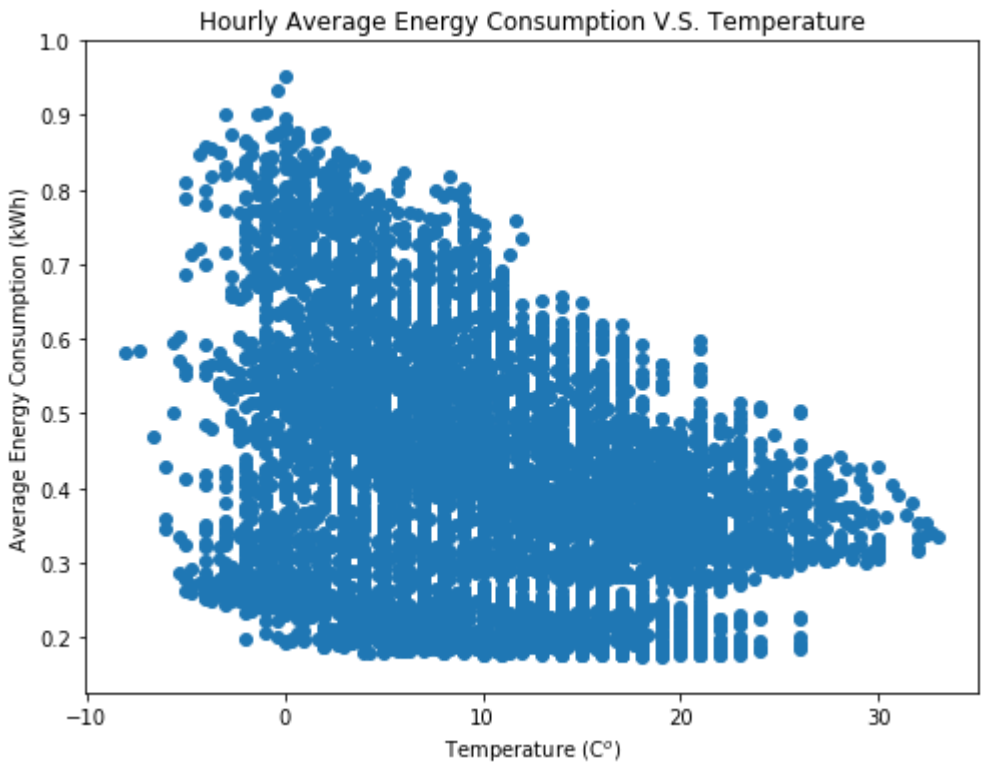
In [4]:
```
# Let's first create two new columns that is the total and average energy consumptions over all users to df_Ntou1h
totalHourly = df_Ntou1h.sum(1)
df_Ntou1h['Total'] = totalHourly
df_Ntou1h['Average'] = totalHourly / (df_Ntou1h.shape[1] - 2) # Delete the first GMT column and the Total column just added
df_Ntou1h.head()
```

Out[4]:

	GMT	N0000	N0001	N0002	N0003	N0004	N0005	N0006	N0007	N0008	...	N4165	N4166	N4167	N4168	N4169	N4170	N4171
0	2013-01-01 00:00:00	1.038	0.232	0.052	0.256	0.181	0.773	0.0	0.163	0.261	...	0.087	0.020	0.586	0.201	0.069	0.254	0.102
1	2013-01-01 01:00:00	0.484	0.267	0.101	0.192	0.185	0.359	0.0	0.152	0.090	...	0.066	0.115	0.464	0.194	0.070	0.163	0.176
2	2013-01-01 02:00:00	0.449	0.266	0.092	0.200	0.484	0.150	0.0	0.170	0.156	...	0.084	0.235	0.190	0.241	0.146	0.097	0.155
3	2013-01-01 03:00:00	0.390	0.230	0.034	0.262	0.132	0.155	0.0	0.152	0.094	...	0.066	0.021	0.157	0.185	0.232	0.137	0.095
4	2013-01-01 04:00:00	0.936	0.268	0.052	0.205	0.183	0.163	0.0	0.165	0.155	...	0.072	0.021	0.175	0.312	0.078	0.089	0.094

5 rows × 4176 columns

In [7]:
```
# from IPython.core.display import HTML
# HTML("""
# <style>
# .output_png {
#     display: flex;
#     text-align: center;
#     vertical-align: center;
#     align-items: center;
# }
# </style>
# """)

plt.figure(figsize = (8, 6))
plt.scatter(df_wea1h.TempC, df_Ntou1h.Average)
plt.xlabel(r'Temperature (C$^o$)')
plt.ylabel('Average Energy Consumption (kWh)')
plt.title('Hourly Average Energy Consumption V.S. Temperature')
plt.show()
```

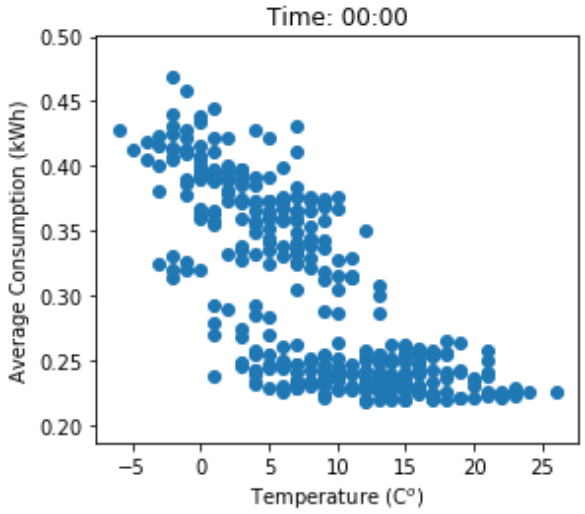


But what if we check the relationship between the average energy consumption and outside temperature at each hour of a day. (for the non-TOU case)

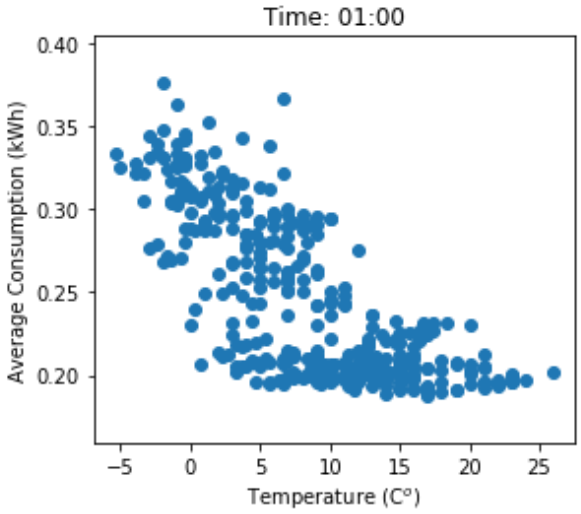
For example, 9:00 am average energy consumption and outside temperature over 365 days.

In [8]:
```
# Show the elements in Average column such that they are hourly energy consumption v.s. temperature over 365
 days
fig_all = plt.figure(figsize = (13,30))
for i in range(24):
    ax = fig_all.add_subplot(8, 3, i + 1)
    if i <= 9:
        ax.scatter(df_wea1h.TempC[df_wea1h.GMT.str.contains('0' + str(i) + ':00:00')], df_Ntou1h.Average[df_N
tou1h.GMT.str.contains('0' + str(i) + ':00:00')] )
        ax.set_title('Time: 0' + str(i) + ':00')
        ax.set_xlabel(r'Temperature (C$^o$)')
        ax.set_ylabel('Average Consumption (kWh)')
    else:
        ax.scatter(df_wea1h.TempC[df_wea1h.GMT.str.contains(str(i) + ':00:00')], df_Ntou1h.Average[df_Ntou1h.
GMT.str.contains(str(i) + ':00:00')] )
        ax.set_title('Time: ' + str(i) + ':00')
        ax.set_xlabel(r'Temperature (C$^o$)')
        ax.set_ylabel('Average Consumption (kWh)')
plt.tight_layout()
```

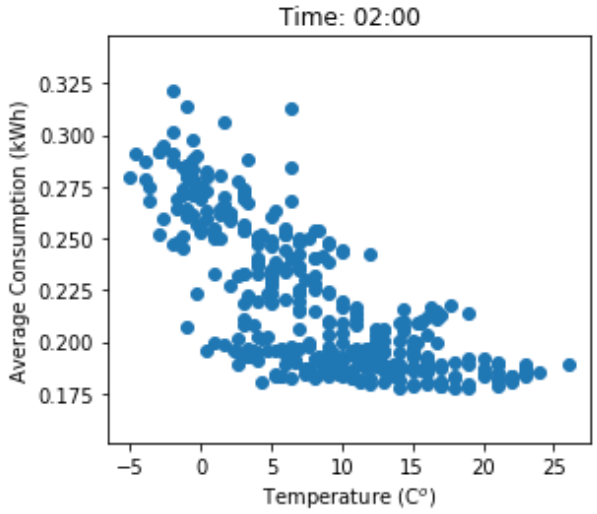
Time: 00:00



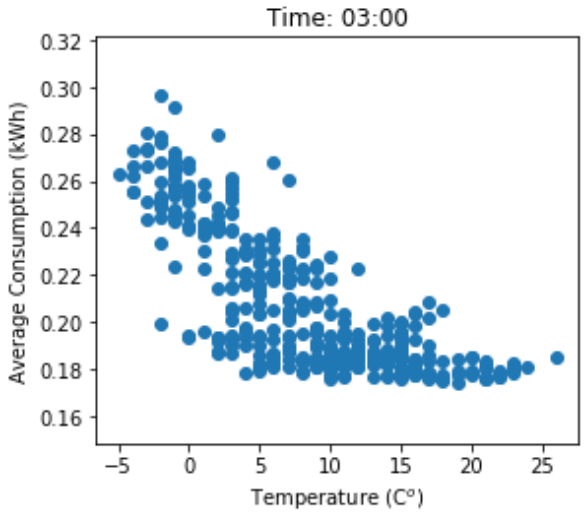
Time: 01:00



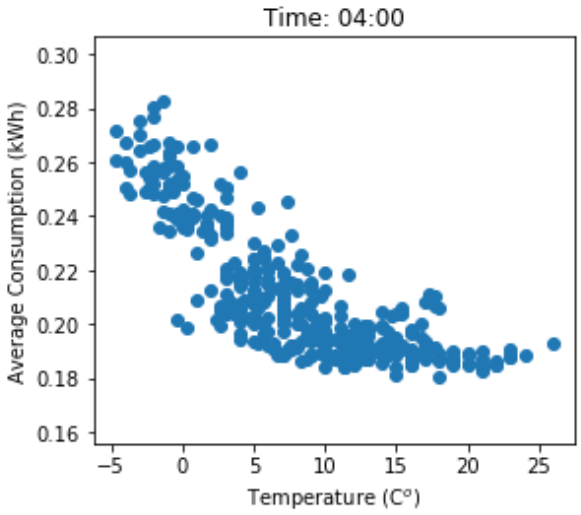
Time: 02:00



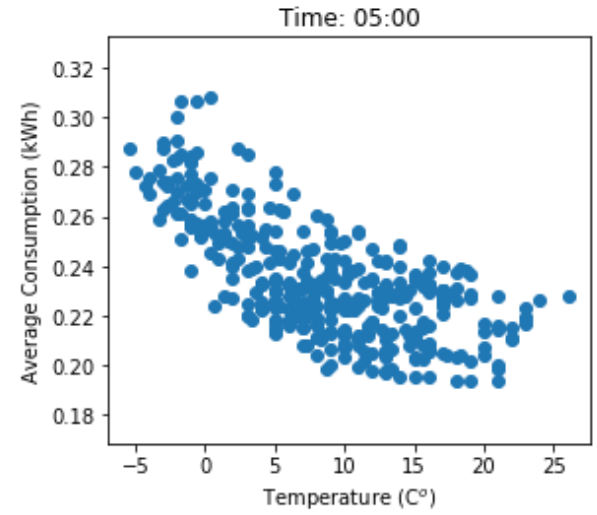
Time: 03:00



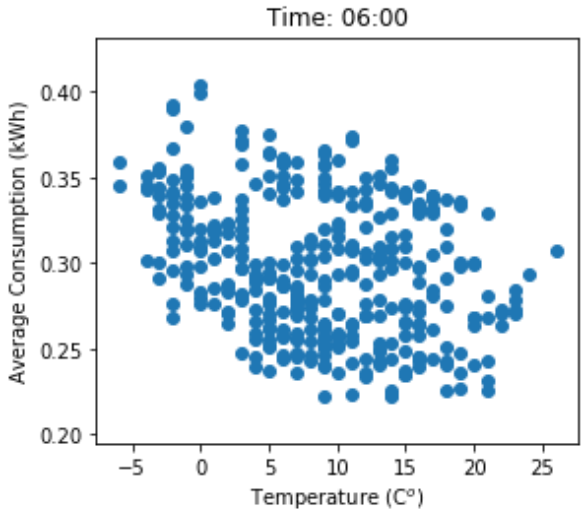
Time: 04:00



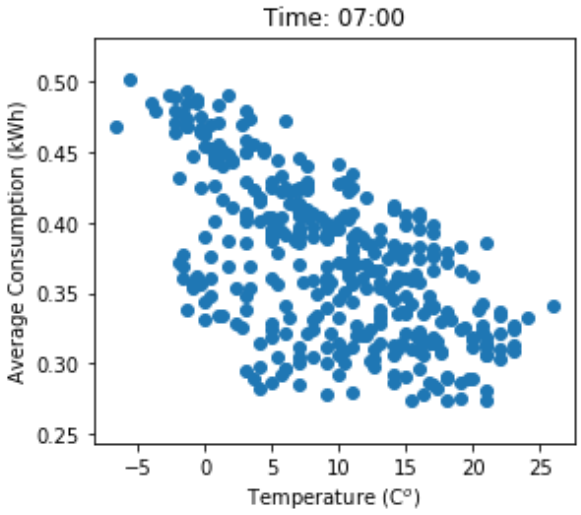
Time: 05:00



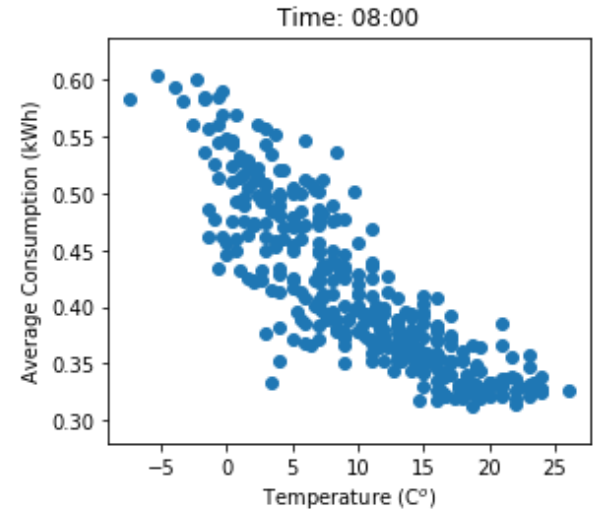
Time: 06:00



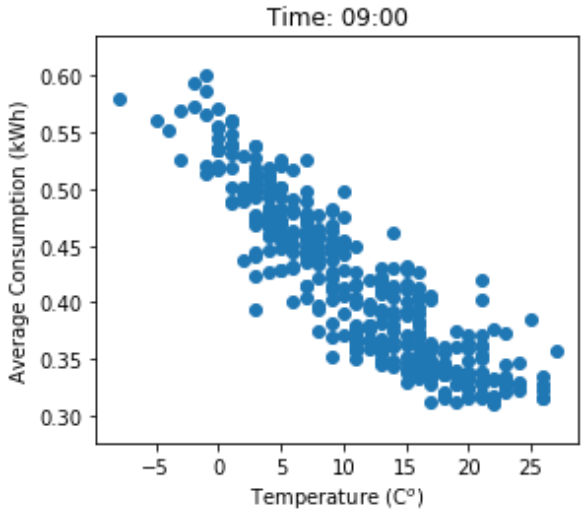
Time: 07:00



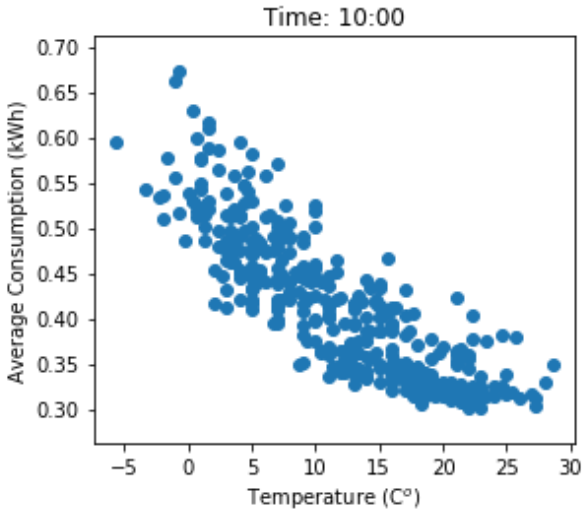
Time: 08:00



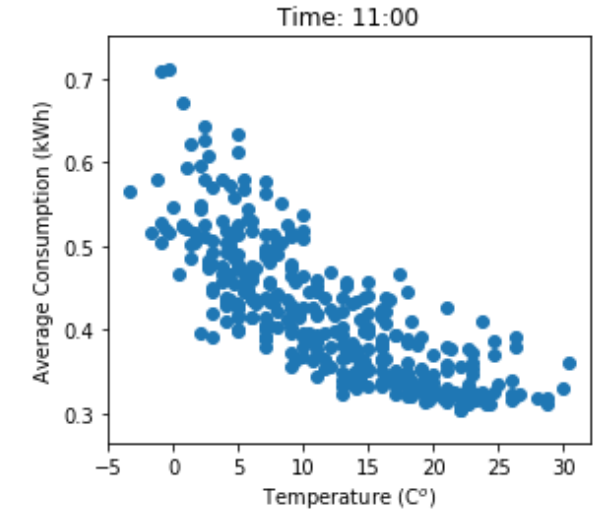
Time: 09:00



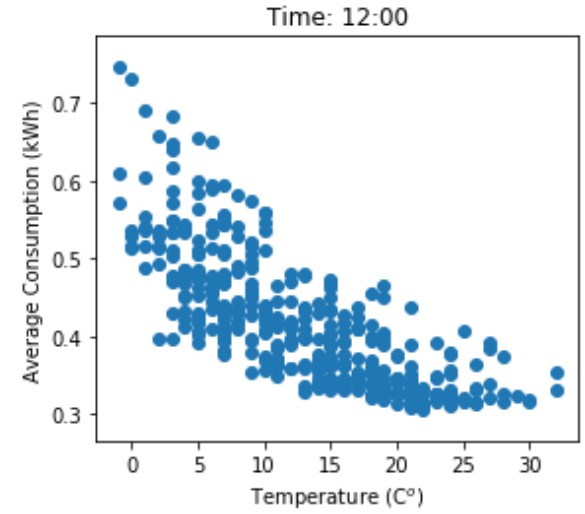
Time: 10:00



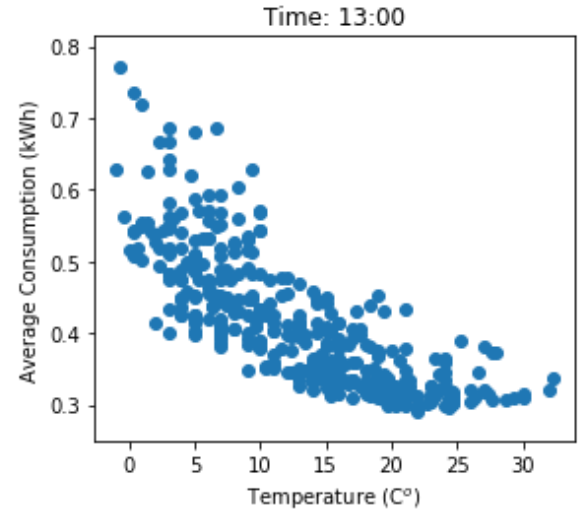
Time: 11:00



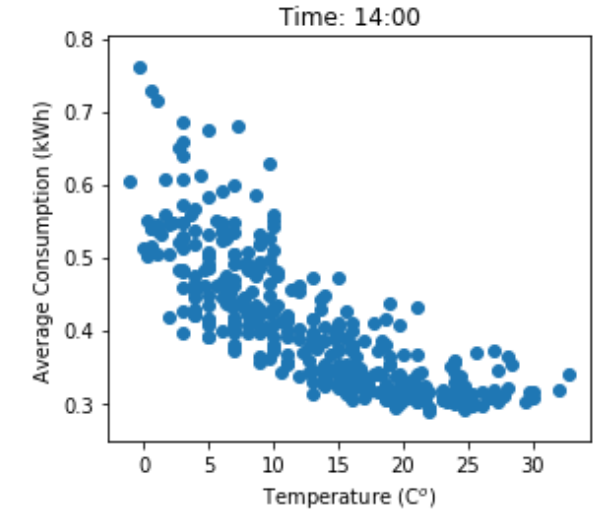
Time: 12:00



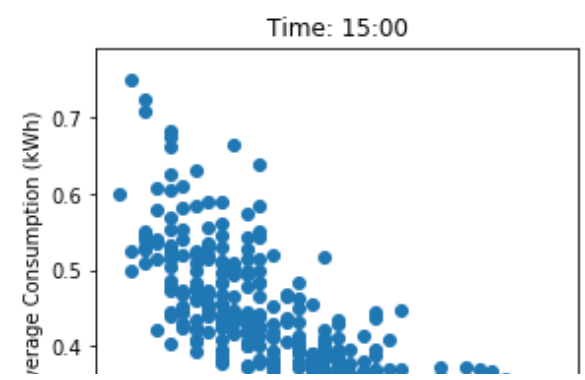
Time: 13:00



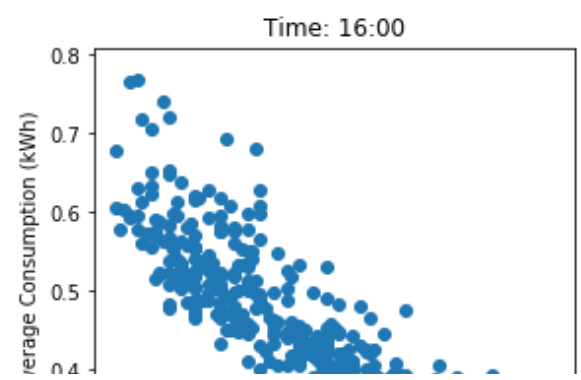
Time: 14:00



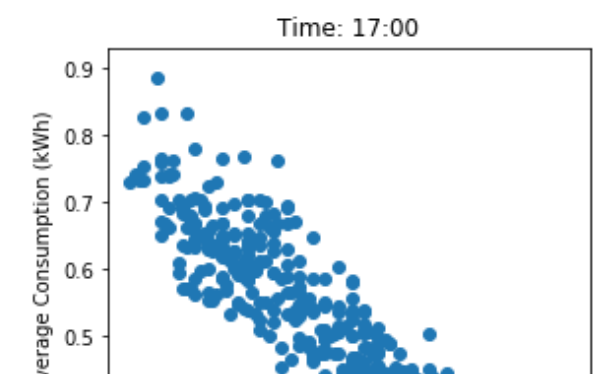
Time: 15:00

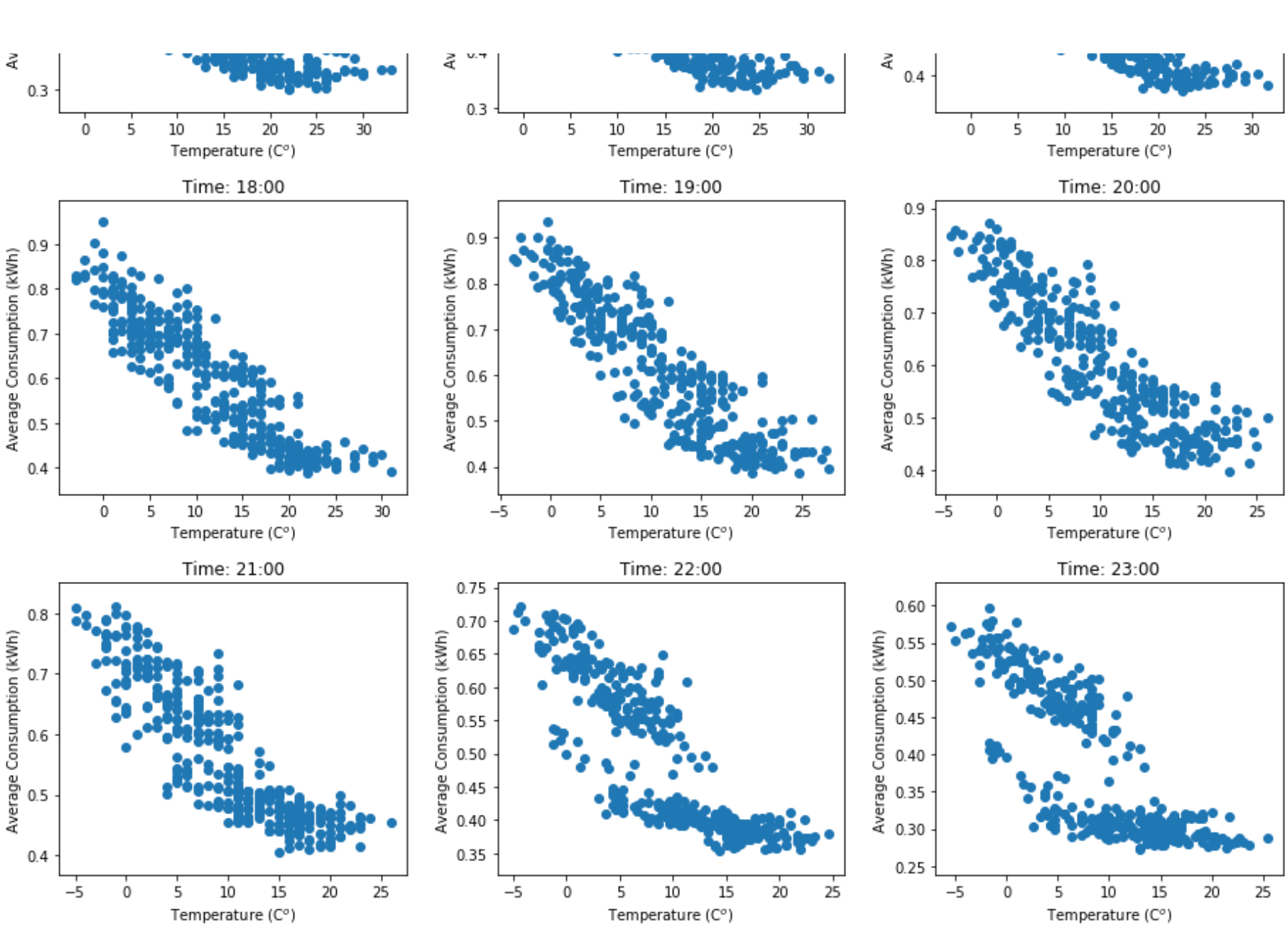


Time: 16:00



Time: 17:00





Let's take a look at the relationship for TOU case

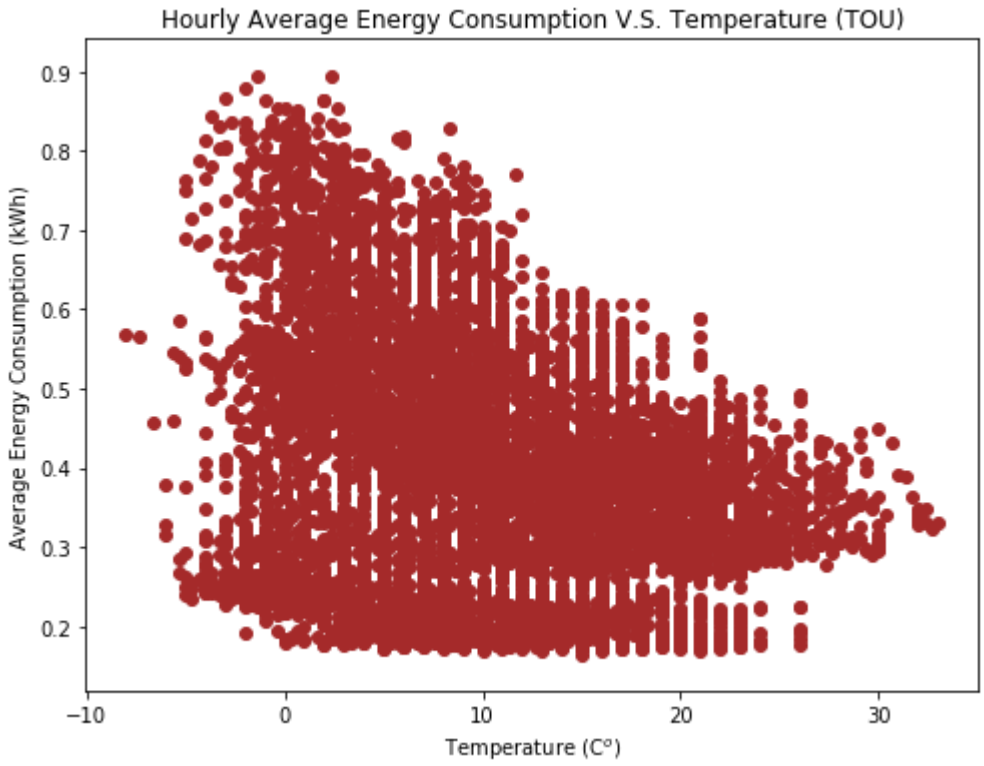
In [5]:
```
# Let's first create two new columns that is the total and average energy consumptions over all users to df_tou1h
totalHourly = df_tou1h.sum(1)
df_tou1h['Total'] = totalHourly
df_tou1h['Average'] = totalHourly / (df_tou1h.shape[1] - 2) # Delete the first GMT column and the Total column just added
df_tou1h.head()
```

Out[5]:

	GMT	D0000	D0001	D0002	D0003	D0004	D0005	D0006	D0007	D0008	...	D1017	D1018	D1019	D1020	D1021	D1022	D1023
0	2013-01-01 00:00:00	1.447	0.429	0.451	0.155	0.397	0.106	0.307	0.268	0.390	...	0.502	0.117	0.378	1.143	0.155	0.343	0.176
1	2013-01-01 01:00:00	0.336	0.377	0.467	0.153	0.227	0.109	0.365	0.300	0.528	...	0.528	0.117	0.399	0.856	0.098	0.278	0.334
2	2013-01-01 02:00:00	0.244	0.151	0.490	0.154	0.126	0.108	0.060	0.197	0.150	...	0.217	0.310	0.362	0.521	0.083	0.088	0.162
3	2013-01-01 03:00:00	0.213	0.155	0.580	0.153	0.057	0.109	0.067	0.167	0.151	...	0.206	0.119	0.368	0.470	0.113	0.061	0.127
4	2013-01-01 04:00:00	0.210	0.121	0.579	0.156	0.061	0.107	0.056	0.144	0.133	...	0.482	0.118	0.354	0.246	0.097	0.058	0.039

5 rows × 1028 columns

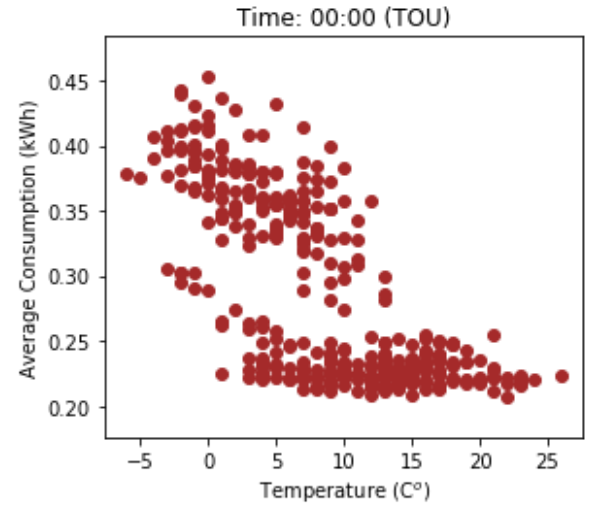
In [10]:
```
plt.figure(figsize = (8, 6))
plt.scatter(df_wea1h.TempC, df_tou1h.Average, c = 'brown')
plt.xlabel(r'Temperature (C$^o$)')
plt.ylabel('Average Energy Consumption (kWh)')
plt.title('Hourly Average Energy Consumption V.S. Temperature (TOU)')
plt.show()
```

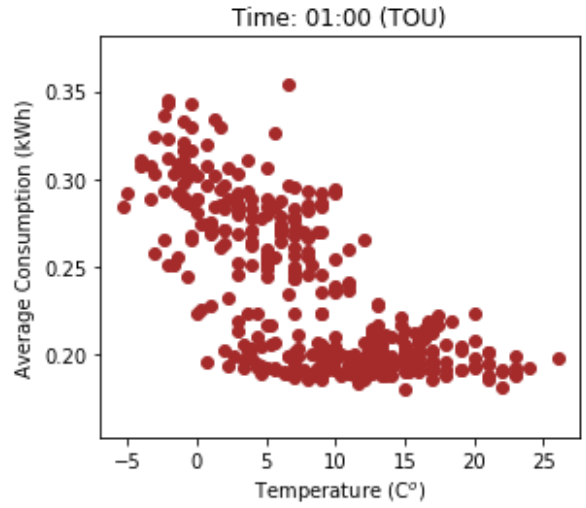


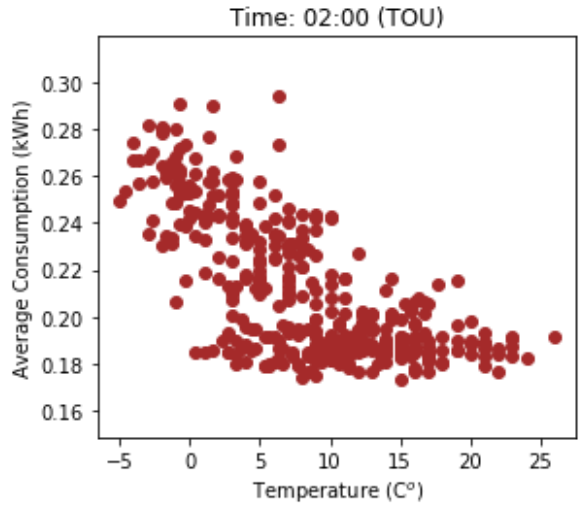
Similarly, if we check the relationship between the average energy consumption and outside temperature at each hour of a day. (for the TOU case)

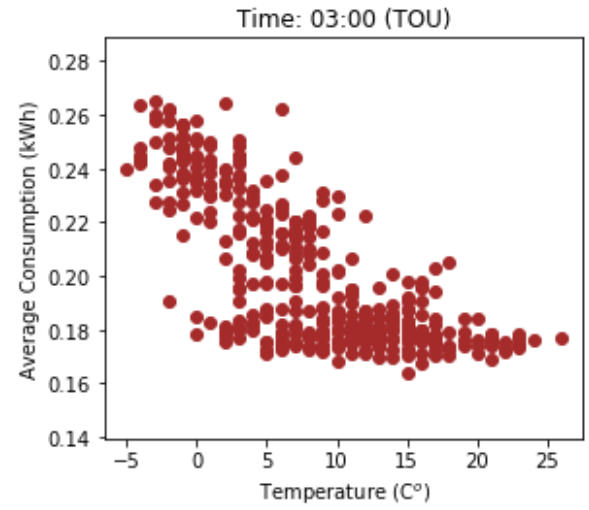
For example, 9:00 am average energy consumption and outside temperature over 365 days.

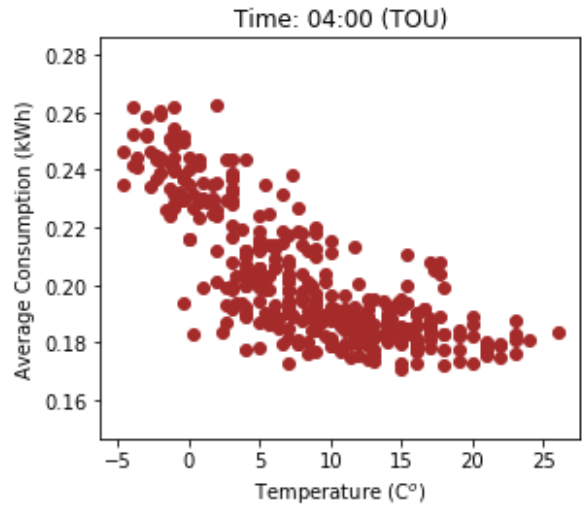
In [11]:
```
# Show the elements in Average column such that they are hourly energy consumption v.s. temperature over 365
 days
fig_all = plt.figure(figsize = (13,30))
for i in range(24):
    ax = fig_all.add_subplot(8, 3, i + 1)
    if i <= 9:
        ax.scatter(df_wea1h.TempC[df_wea1h.GMT.str.contains('0' + str(i) + ':00:00')], df_tou1h.Average[df_to
u1h.GMT.str.contains('0' + str(i) + ':00:00')],  c = 'brown' )
        ax.set_title('Time: 0' + str(i) + ':00 (TOU)')
        ax.set_xlabel(r'Temperature (C$^o$)')
        ax.set_ylabel('Average Consumption (kWh)')
    else:
        ax.scatter(df_wea1h.TempC[df_wea1h.GMT.str.contains(str(i) + ':00:00')], df_tou1h.Average[df_tou1h.GM
T.str.contains(str(i) + ':00:00')], c = 'brown' )
        ax.set_title('Time: ' + str(i) + ':00 (TOU)')
        ax.set_xlabel(r'Temperature (C$^o$)')
        ax.set_ylabel('Average Consumption (kWh)')
plt.tight_layout()
```

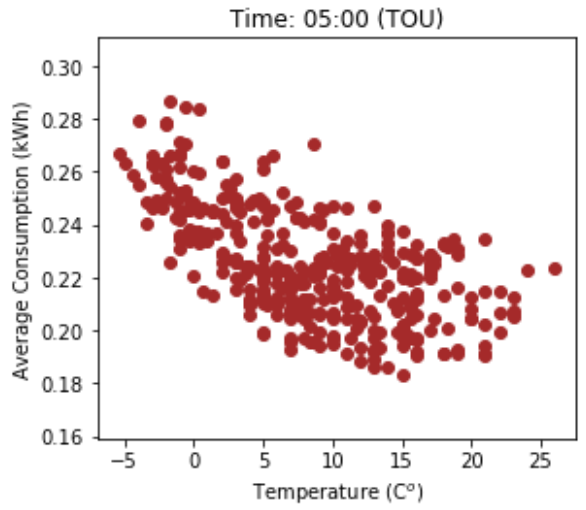

Time: 00:00 (TOU)
Average Consumption (kWh)
Temperature (C°)

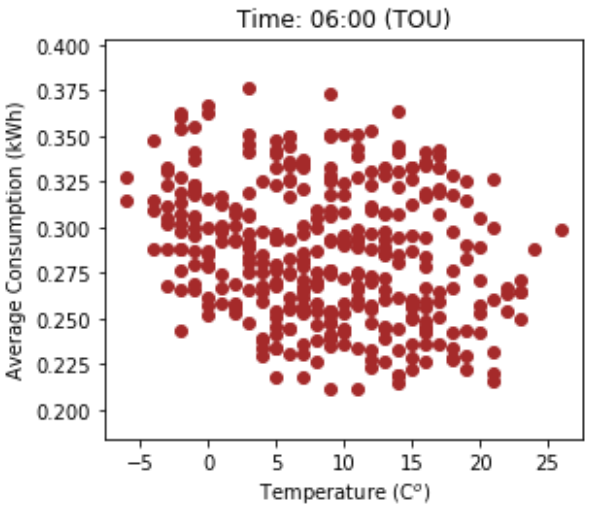

Time: 01:00 (TOU)
Average Consumption (kWh)
Temperature (C°)

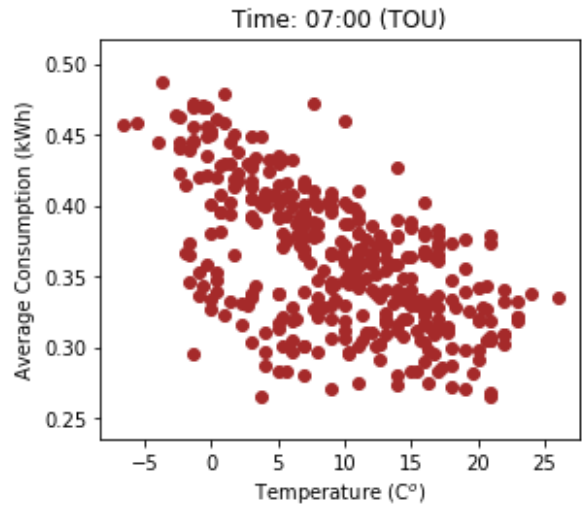

Time: 02:00 (TOU)
Average Consumption (kWh)
Temperature (C°)

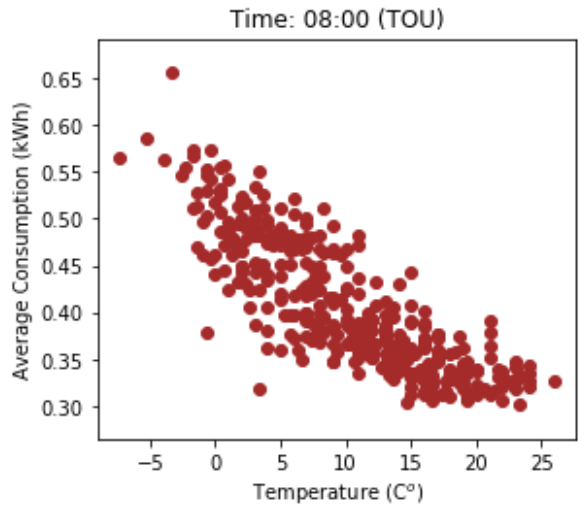

Time: 03:00 (TOU)
Average Consumption (kWh)
Temperature (C°)

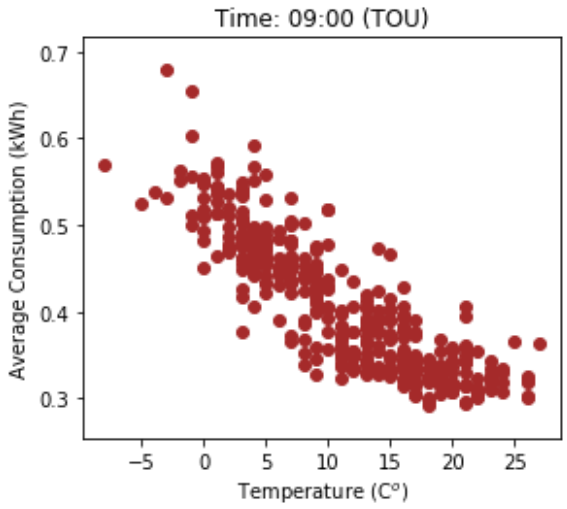

Time: 04:00 (TOU)
Average Consumption (kWh)
Temperature (C°)

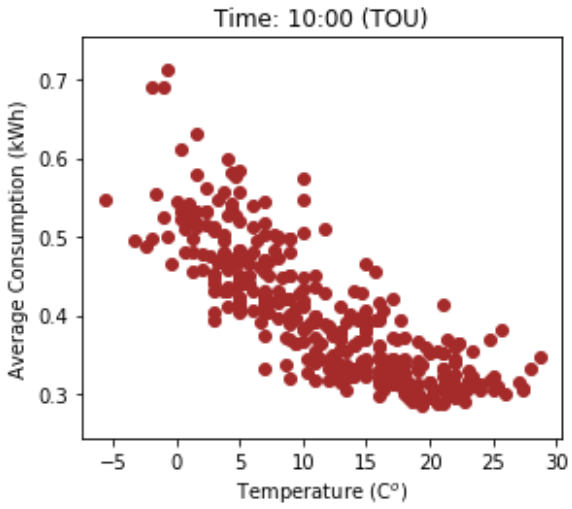

Time: 05:00 (TOU)
Average Consumption (kWh)
Temperature (C°)

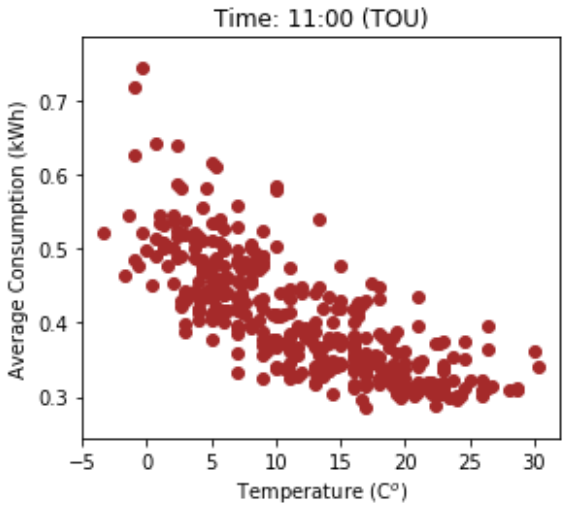

Time: 06:00 (TOU)
Average Consumption (kWh)
Temperature (C°)

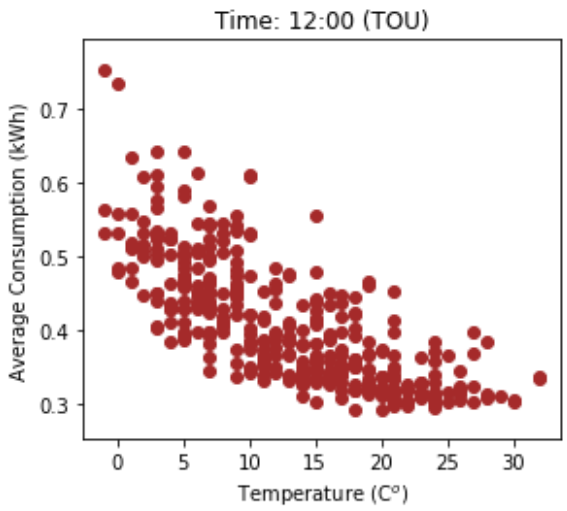

Time: 07:00 (TOU)
Average Consumption (kWh)
Temperature (C°)

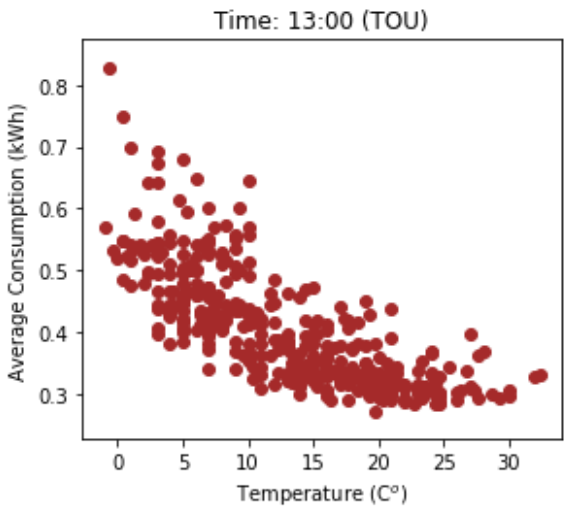

Time: 08:00 (TOU)
Average Consumption (kWh)
Temperature (C°)

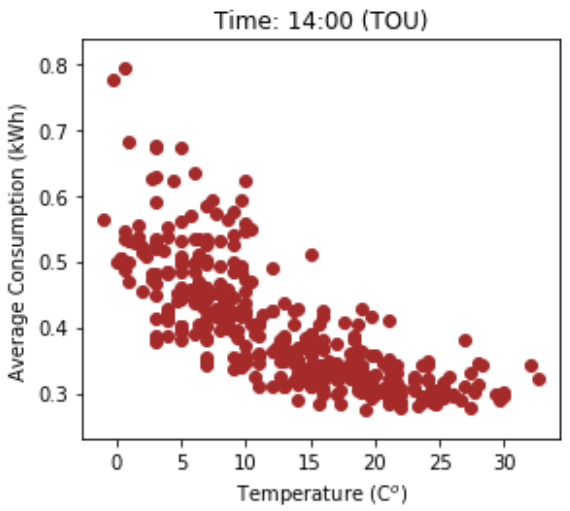

Time: 09:00 (TOU)
Average Consumption (kWh)
Temperature (C°)

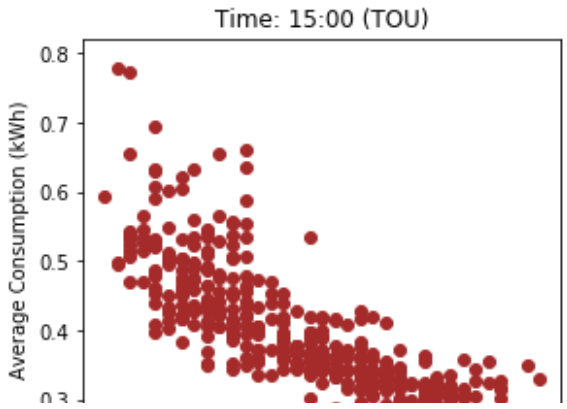

Time: 10:00 (TOU)
Average Consumption (kWh)
Temperature (C°)

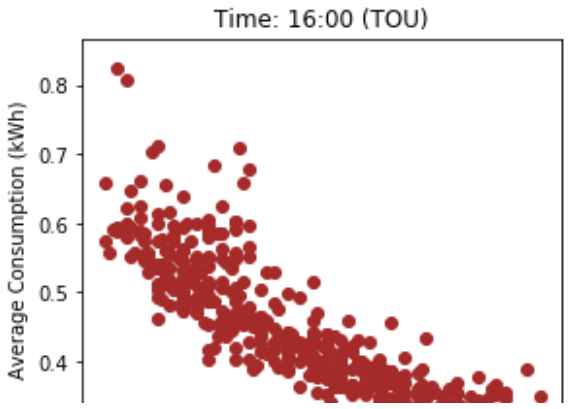

Time: 11:00 (TOU)
Average Consumption (kWh)
Temperature (C°)

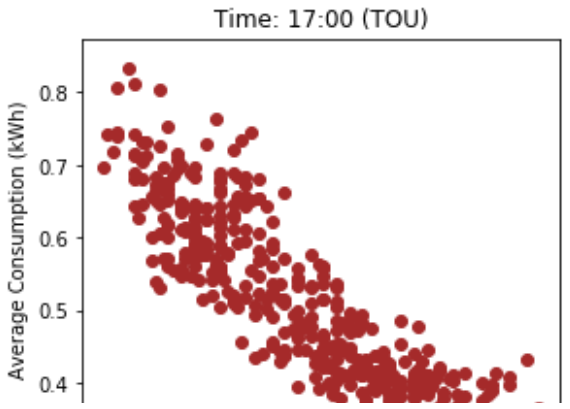

Time: 12:00 (TOU)
Average Consumption (kWh)
Temperature (C°)

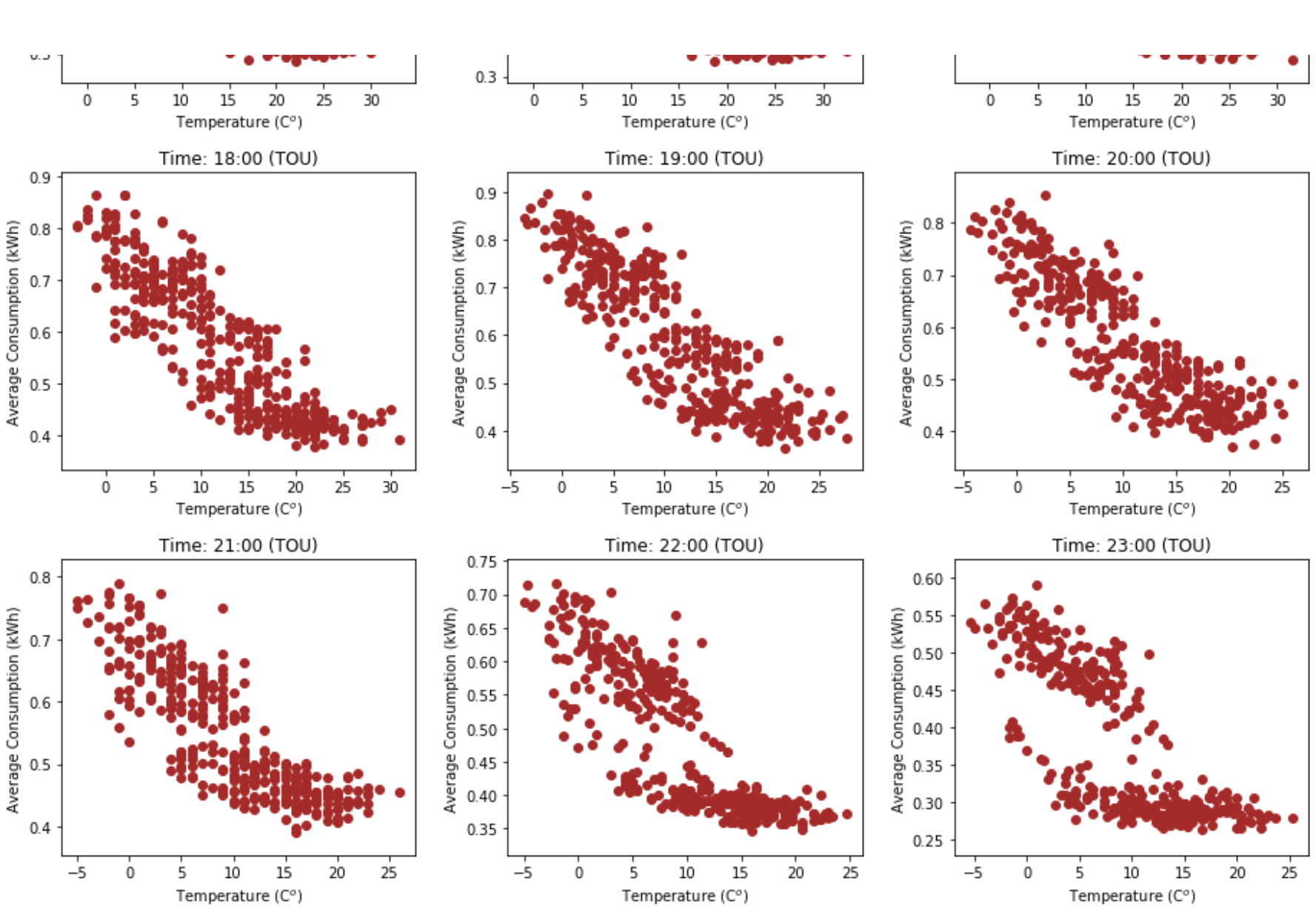

Time: 13:00 (TOU)
Average Consumption (kWh)
Temperature (C°)


Time: 14:00 (TOU)
Average Consumption (kWh)
Temperature (C°)


Time: 15:00 (TOU)
Average Consumption (kWh)

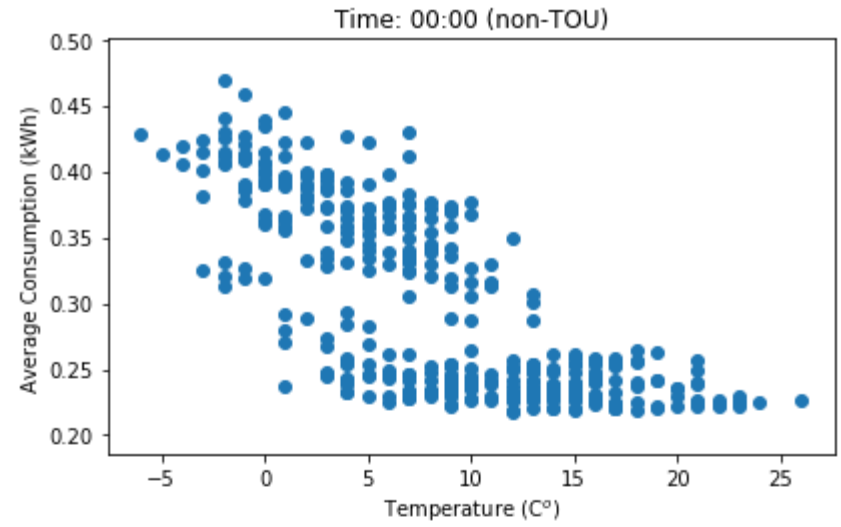

Time: 16:00 (TOU)
Average Consumption (kWh)

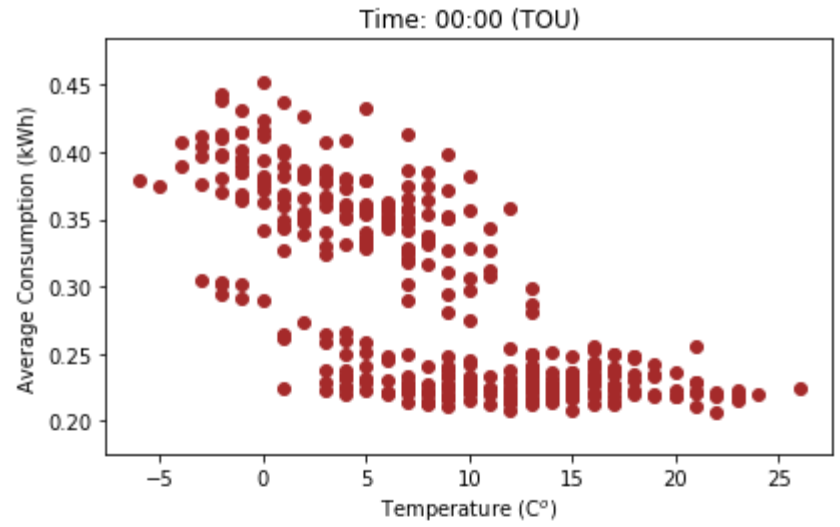

Time: 17:00 (TOU)
Average Consumption (kWh)

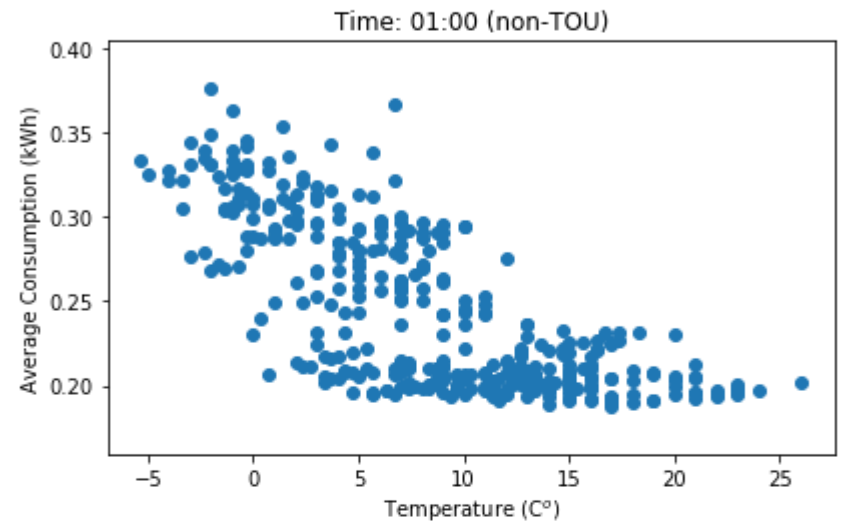


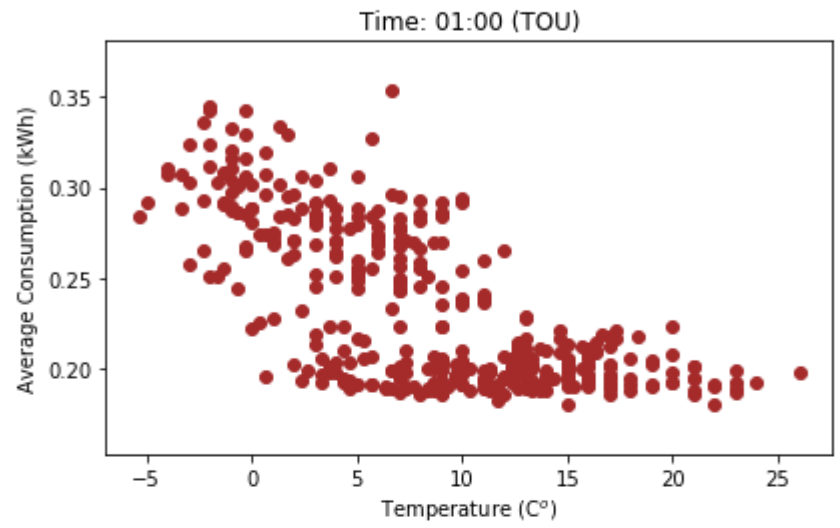
Let's compare these two cases, i.e. non-TOU vs TOU

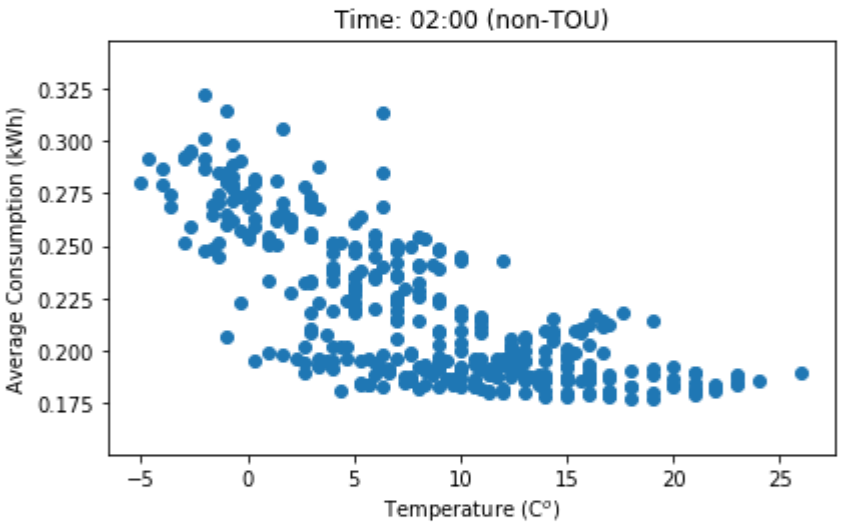
In [12]:
```
# non-TOU vs TOU in terms of showing the elements in Average column such that they are hourly energy consumpt
ion v.s. temperature over 365 days
fig_all = plt.figure(figsize = (12,90))
ax_Ntou = [] # store subplot objects
ax_tou = [] # store subplot objects
for i in range(24):
    ax_Ntou.append(fig_all.add_subplot(24, 2, 2 * i + 1))
    if i <= 9:
        ax_Ntou[-1].scatter(df_wea1h.TempC[df_wea1h.GMT.str.contains('0' + str(i) + ':00:00')], df_Ntou1h.Ave
rage[df_Ntou1h.GMT.str.contains('0' + str(i) + ':00:00')] )
        ax_Ntou[-1].set_title('Time: 0' + str(i) + ':00 (non-TOU)')
        ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_Ntou[-1].set_ylabel('Average Consumption (kWh)')
    else:
        ax_Ntou[-1].scatter(df_wea1h.TempC[df_wea1h.GMT.str.contains(str(i) + ':00:00')], df_Ntou1h.Average[d
f_Ntou1h.GMT.str.contains(str(i) + ':00:00')] )
        ax_Ntou[-1].set_title('Time: ' + str(i) + ':00 (non-TOU)')
        ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_Ntou[-1].set_ylabel('Average Consumption (kWh)')
for i in range(24):
    ax_tou.append(fig_all.add_subplot(24, 2, 2 * i + 2))
    if i <= 9:
        ax_tou[-1].scatter(df_wea1h.TempC[df_wea1h.GMT.str.contains('0' + str(i) + ':00:00')], df_tou1h.Avera
ge[df_tou1h.GMT.str.contains('0' + str(i) + ':00:00')],  c = 'brown' )
        ax_tou[-1].set_title('Time: 0' + str(i) + ':00 (TOU)')
        ax_tou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_tou[-1].set_ylabel('Average Consumption (kWh)')
    else:
        ax_tou[-1].scatter(df_wea1h.TempC[df_wea1h.GMT.str.contains(str(i) + ':00:00')], df_tou1h.Average[df_
tou1h.GMT.str.contains(str(i) + ':00:00')], c = 'brown' )
        ax_tou[-1].set_title('Time: ' + str(i) + ':00 (TOU)')
        ax_tou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_tou[-1].set_ylabel('Average Consumption (kWh)')
plt.tight_layout()
```

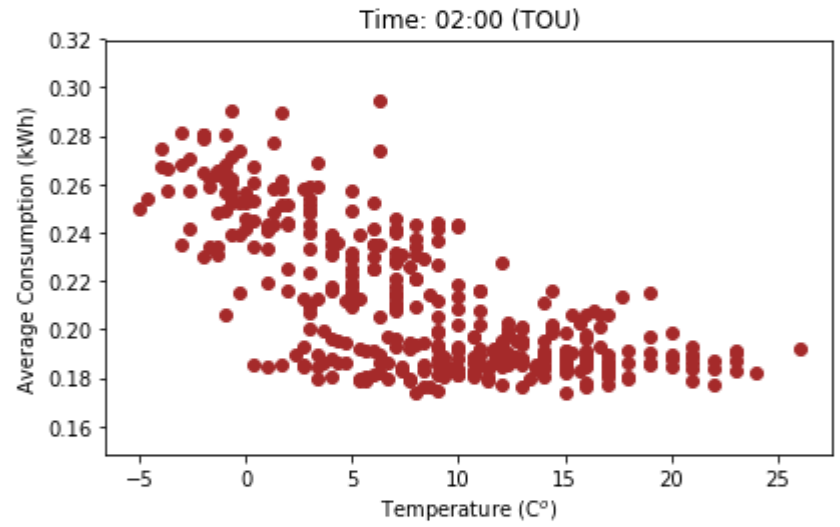

Time: 00:00 (non-TOU)
Average Consumption (kWh)
Temperature (C°)

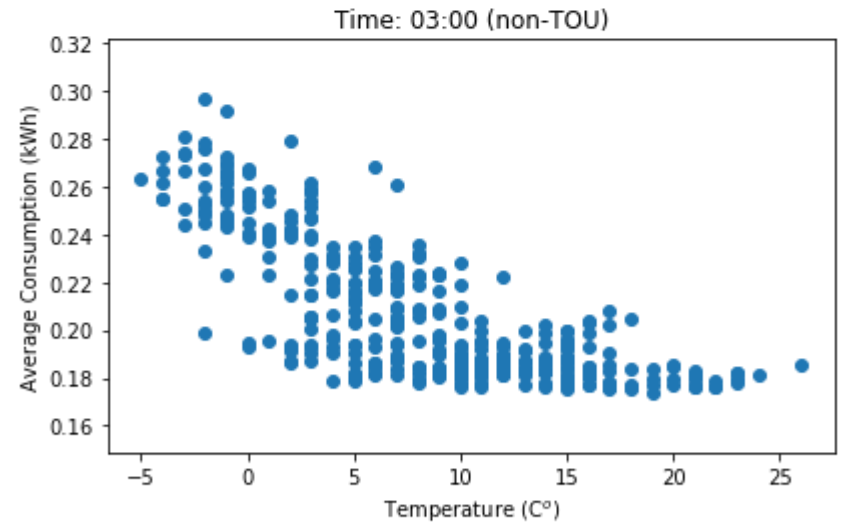

Time: 00:00 (TOU)
Average Consumption (kWh)
Temperature (C°)

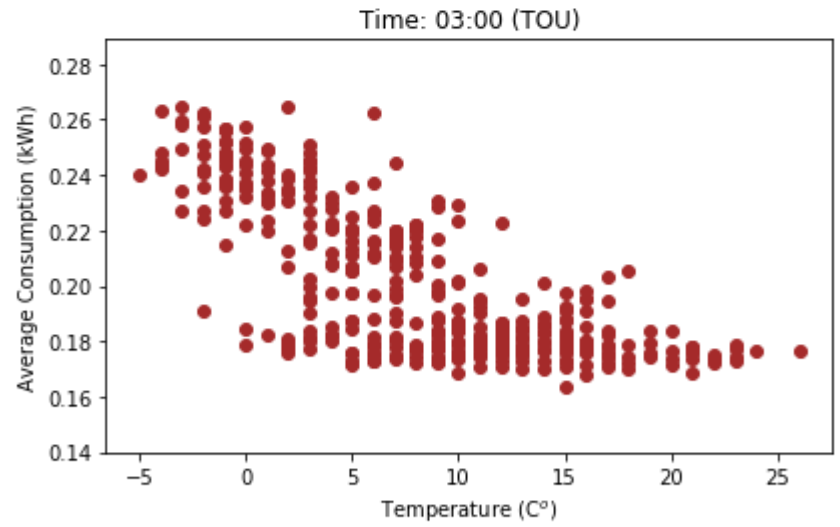

Time: 01:00 (non-TOU)
Average Consumption (kWh)
Temperature (C°)

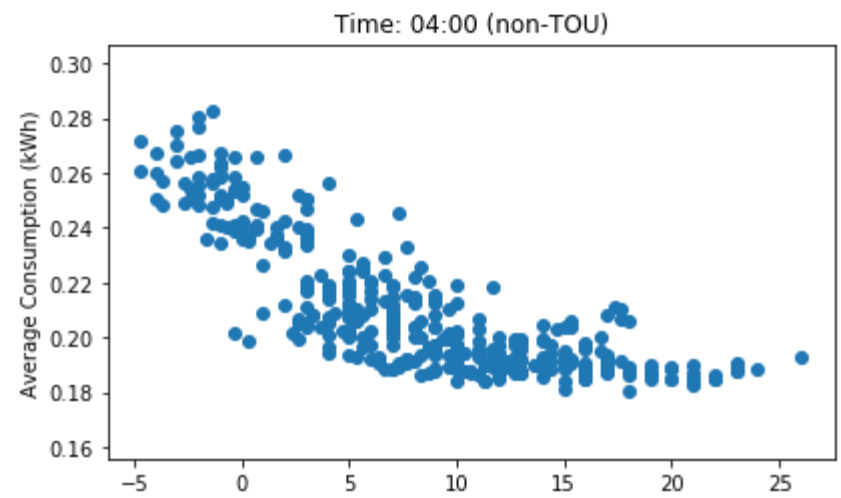

Time: 01:00 (TOU)
Average Consumption (kWh)
Temperature (C°)

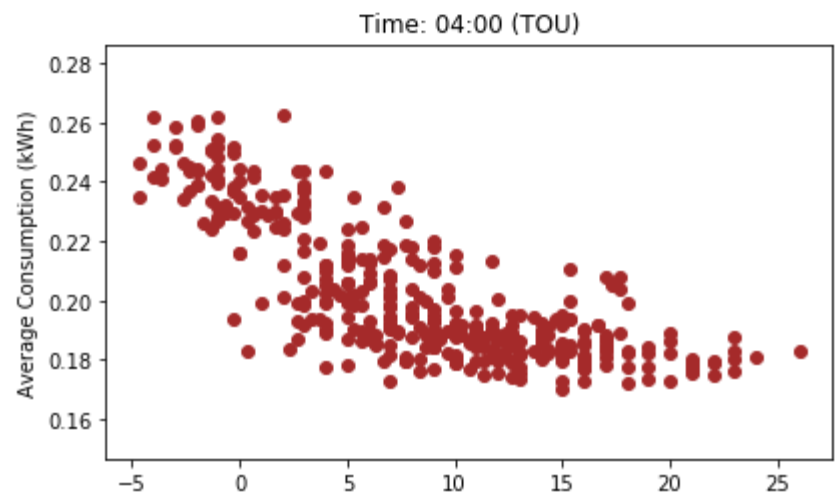

Time: 02:00 (non-TOU)
Average Consumption (kWh)
Temperature (C°)


Time: 02:00 (TOU)
Average Consumption (kWh)
Temperature (C°)


Time: 03:00 (non-TOU)
Average Consumption (kWh)
Temperature (C°)

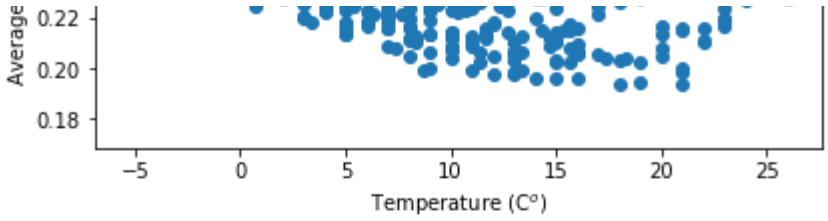

Time: 03:00 (TOU)
Average Consumption (kWh)
Temperature (C°)

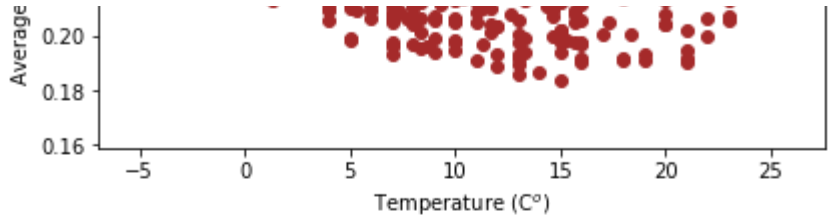

Time: 04:00 (non-TOU)
Average Consumption (kWh)
Temperature (C°)

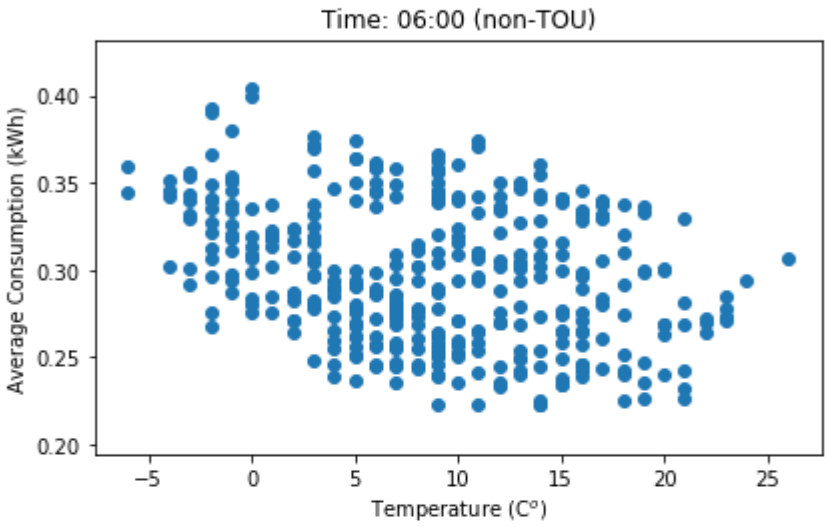

Time: 04:00 (TOU)
Average Consumption (kWh)
Temperature (C°)

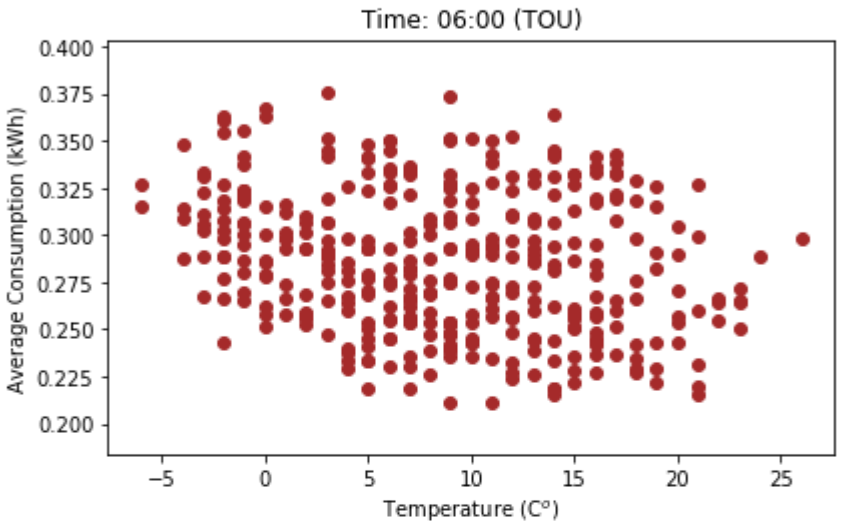
Time: 05:00 (non-TOU)
Temperature (C°)

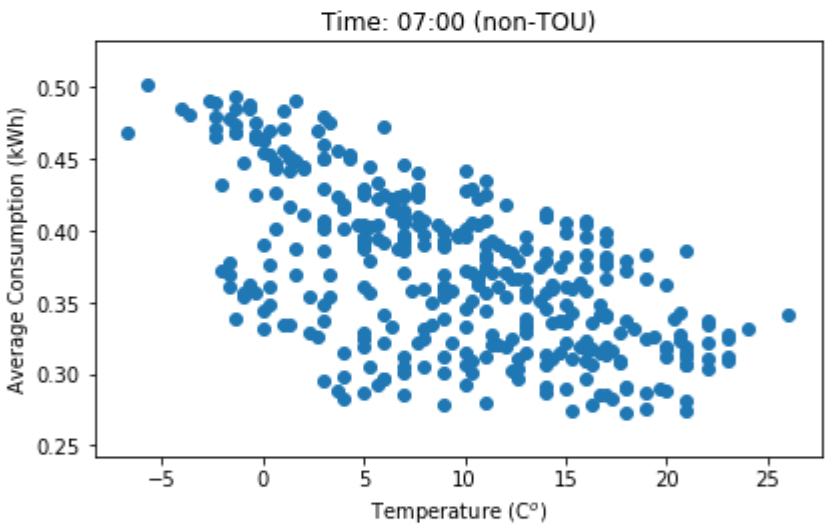
Time: 05:00 (TOU)
Temperature (C°)

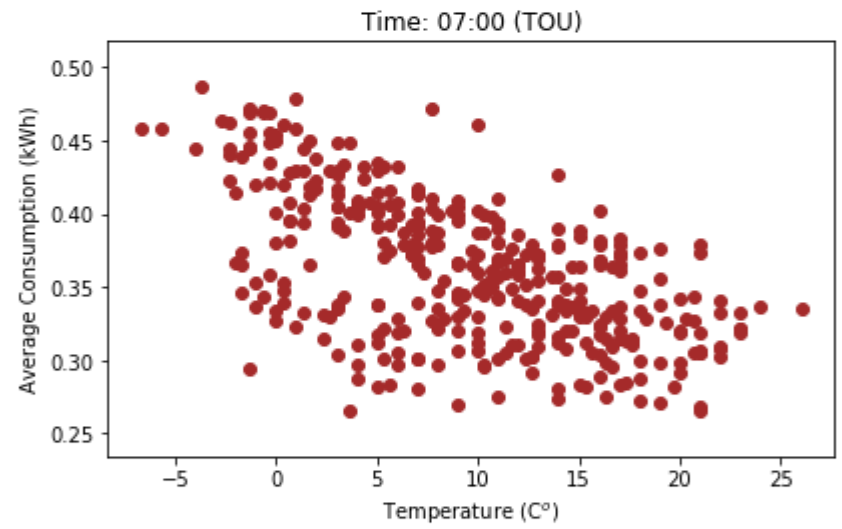

Average
0.22
0.20
0.18
−5
0
5
10
15
20
25
Temperature (C°)

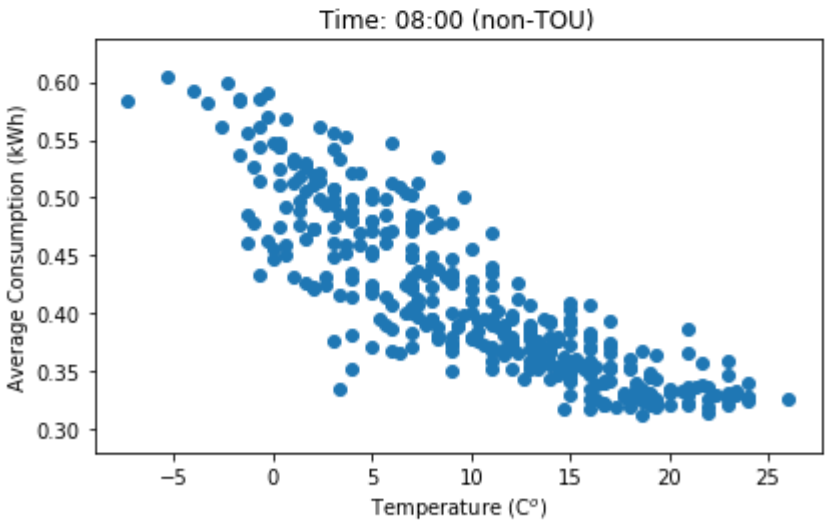

Average
0.20
0.18
0.16
−5
0
5
10
15
20
25
Temperature (C°)

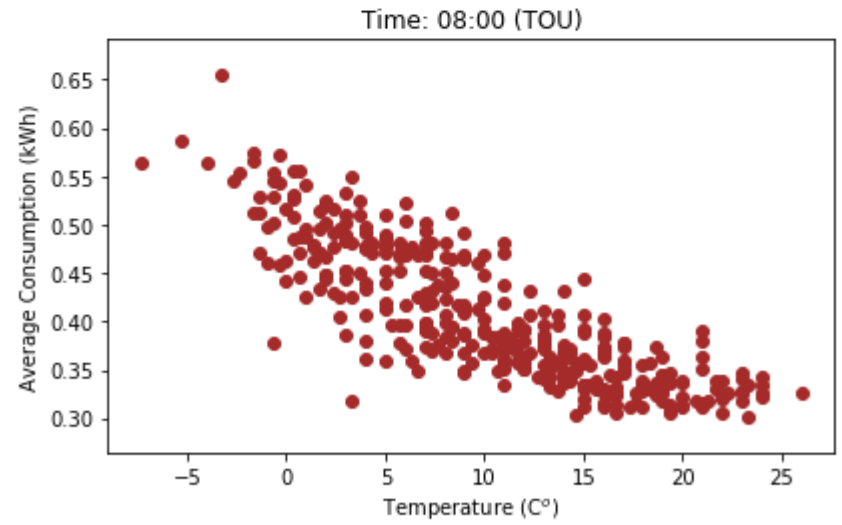

Time: 06:00 (non-TOU)
Average Consumption (kWh)
0.40
0.35
0.30
0.25
0.20
−5
0
5
10
15
20
25
Temperature (C°)

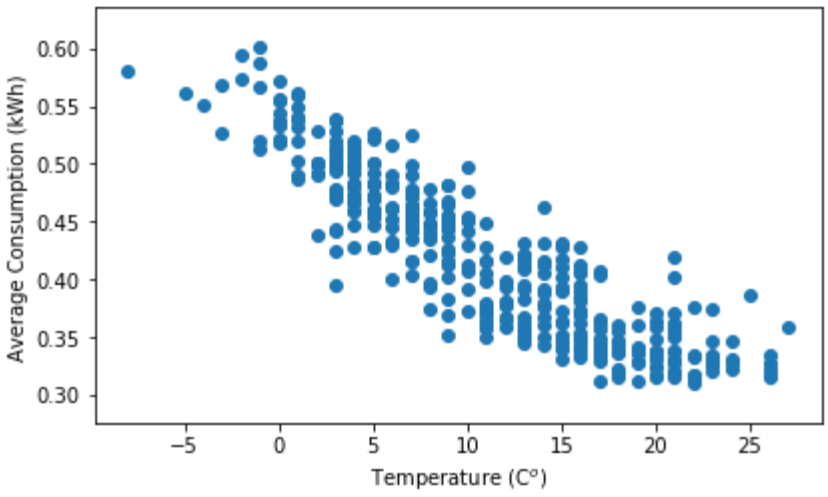

Time: 06:00 (TOU)
Average Consumption (kWh)
0.400
0.375
0.350
0.325
0.300
0.275
0.250
0.225
0.200
−5
0
5
10
15
20
25
Temperature (C°)

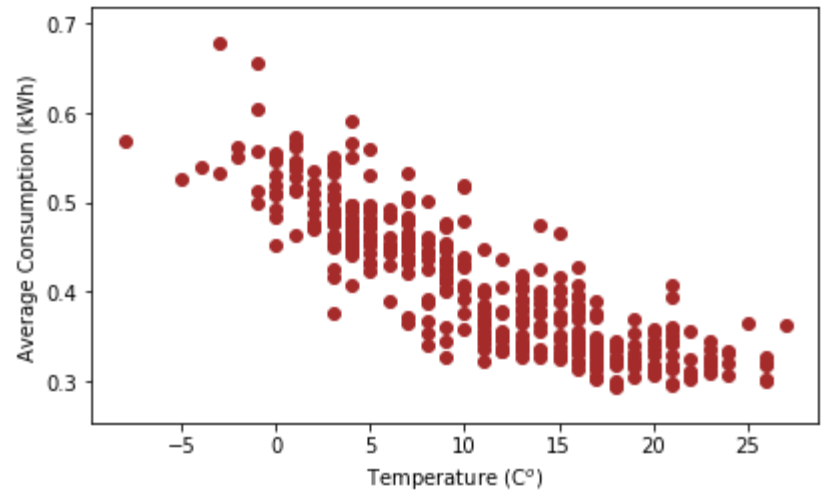

Time: 07:00 (non-TOU)
Average Consumption (kWh)
0.50
0.45
0.40
0.35
0.30
0.25
−5
0
5
10
15
20
25
Temperature (C°)

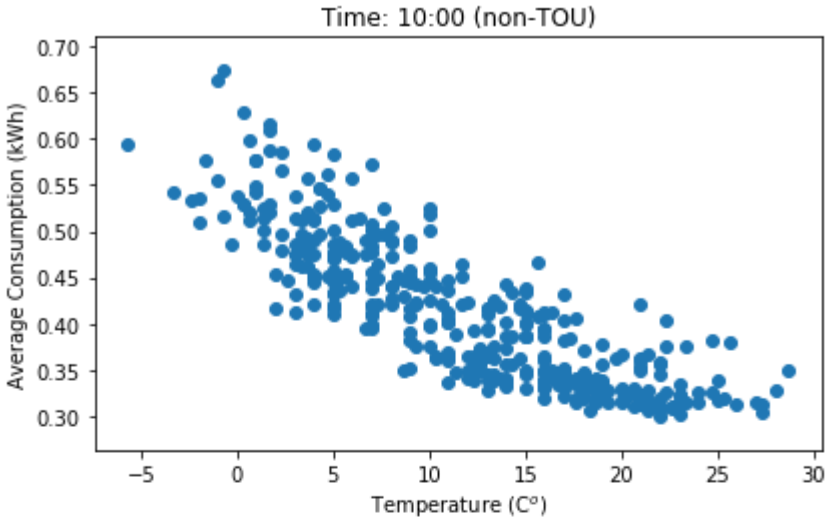

Time: 07:00 (TOU)
Average Consumption (kWh)
0.50
0.45
0.40
0.35
0.30
0.25
−5
0
5
10
15
20
25
Temperature (C°)

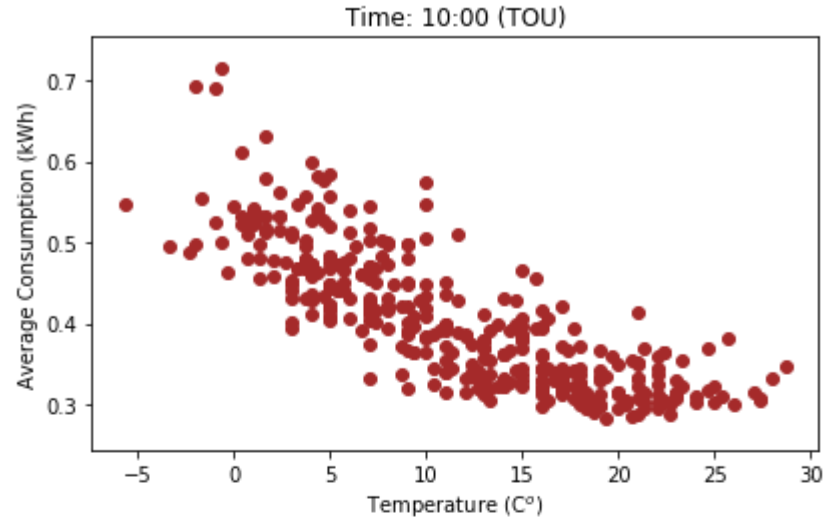

Time: 08:00 (non-TOU)
Average Consumption (kWh)
0.60
0.55
0.50
0.45
0.40
0.35
0.30
−5
0
5
10
15
20
25
Temperature (C°)

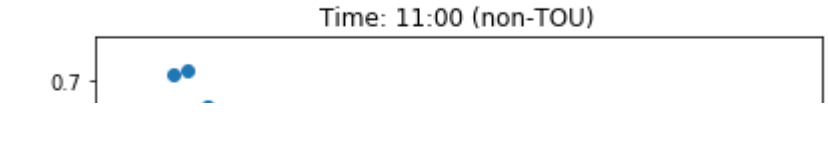

Time: 08:00 (TOU)
Average Consumption (kWh)
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
−5
0
5
10
15
20
25
Temperature (C°)

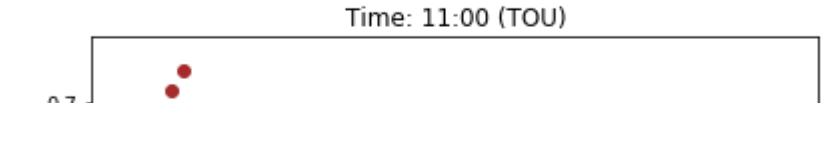

Time: 09:00 (non-TOU)
Average Consumption (kWh)
0.60
0.55
0.50
0.45
0.40
0.35
0.30
−5
0
5
10
15
20
25
Temperature (C°)

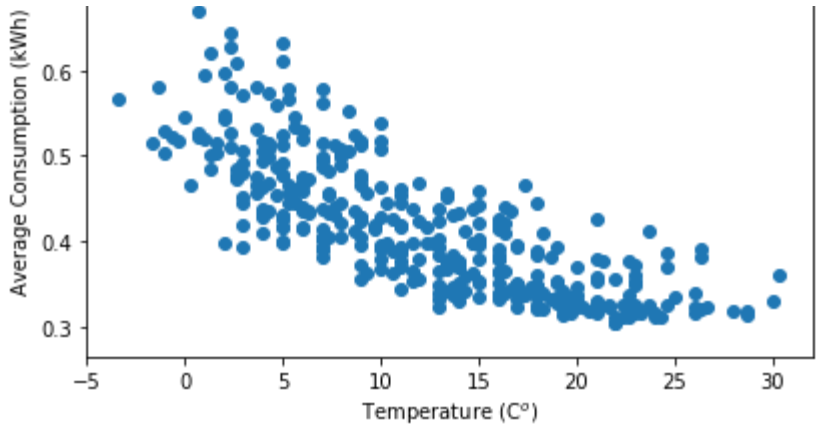

Time: 09:00 (TOU)
Average Consumption (kWh)
0.7
0.6
0.5
0.4
0.3
−5
0
5
10
15
20
25
Temperature (C°)

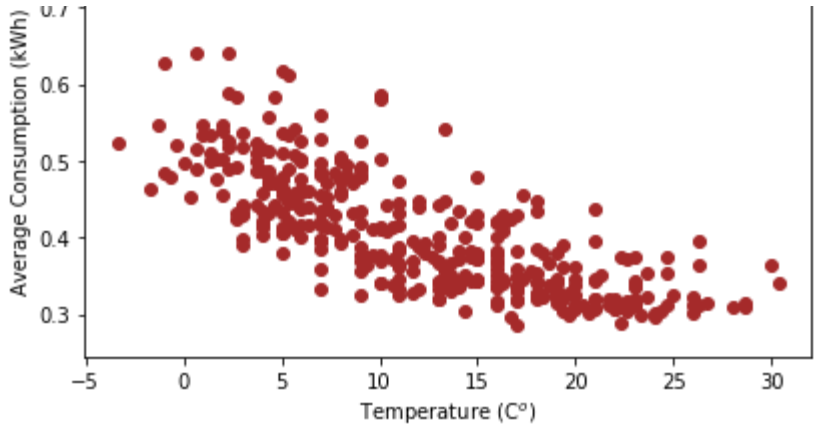

Time: 10:00 (non-TOU)
Average Consumption (kWh)
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
−5
0
5
10
15
20
25
30
Temperature (C°)

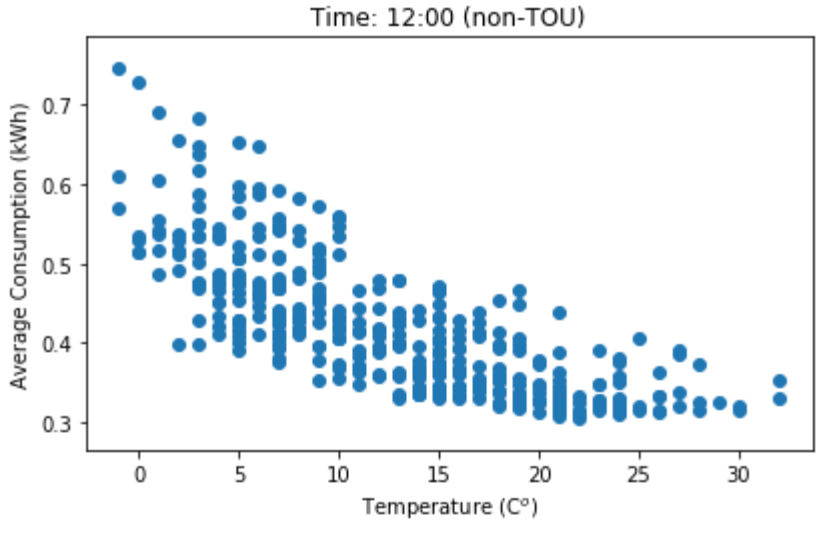

Time: 10:00 (TOU)
Average Consumption (kWh)
0.7
0.6
0.5
0.4
0.3
−5
0
5
10
15
20
25
30
Temperature (C°)

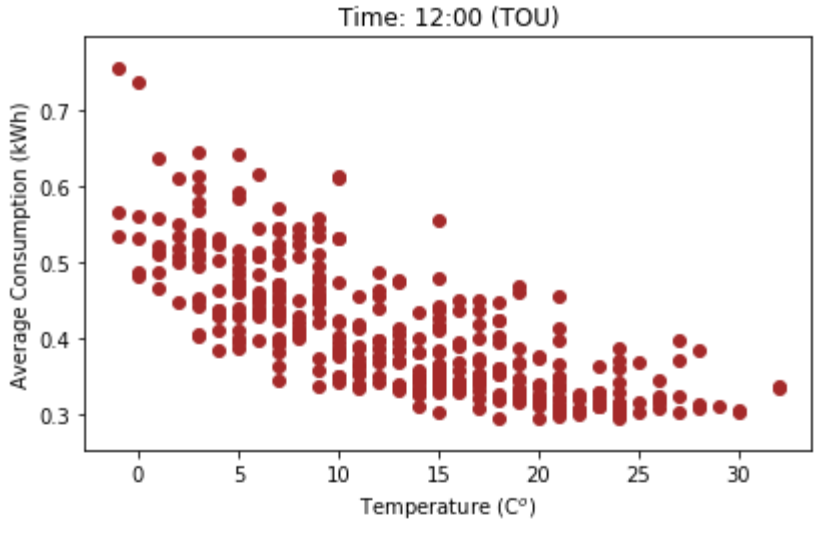

Time: 11:00 (non-TOU)
0.7

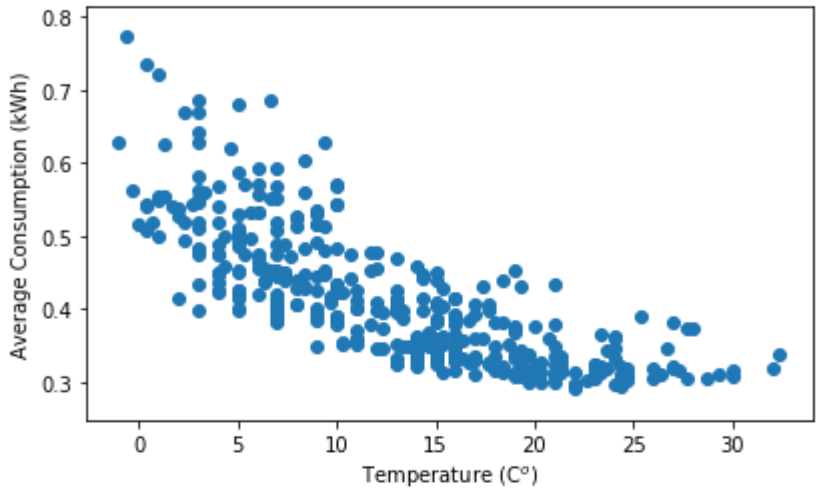

Time: 11:00 (TOU)

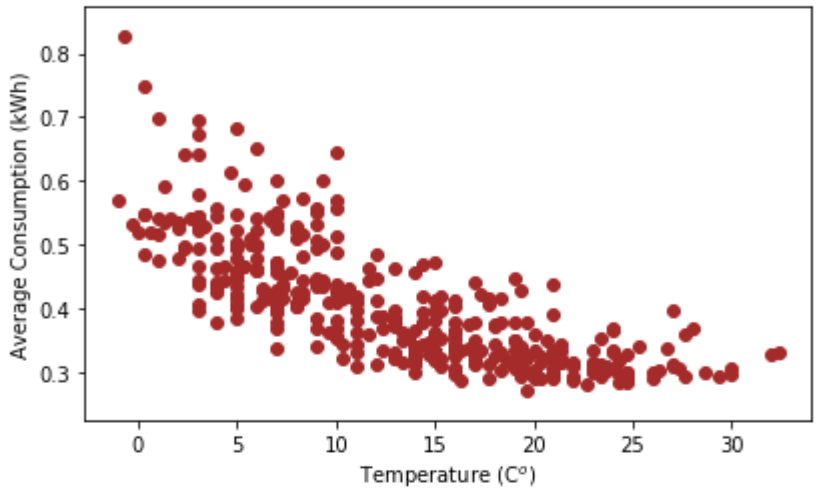

Average Consumption (kWh)
Temperature (C°)

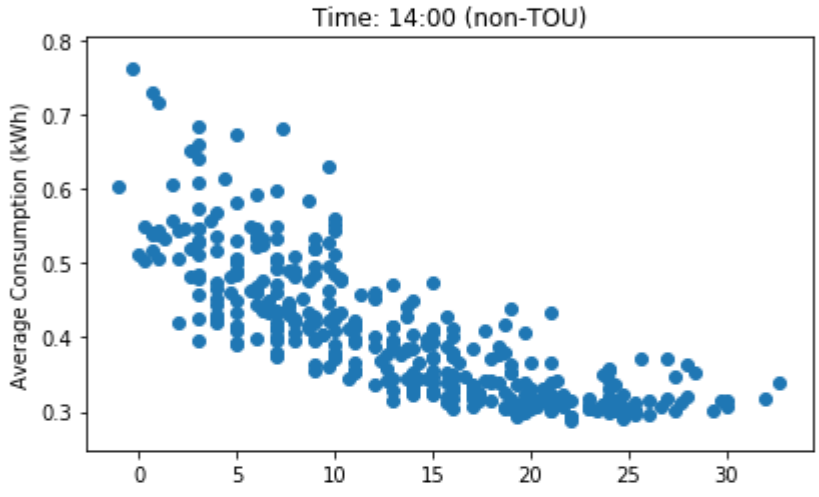

Average Consumption (kWh)
Temperature (C°)

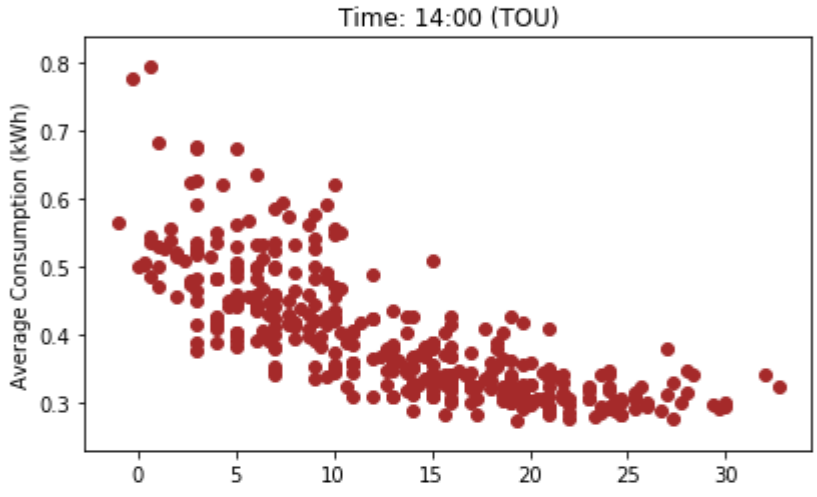

Time: 12:00 (non-TOU)
Average Consumption (kWh)
Temperature (C°)

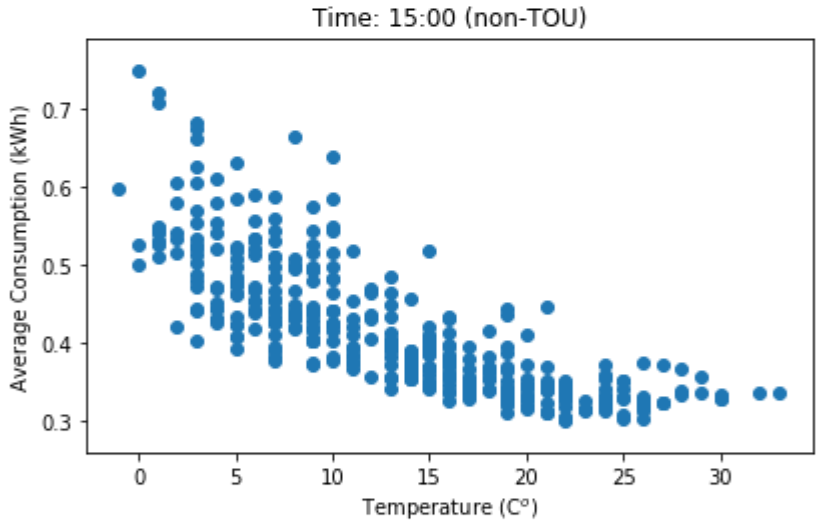

Time: 12:00 (TOU)
Average Consumption (kWh)
Temperature (C°)

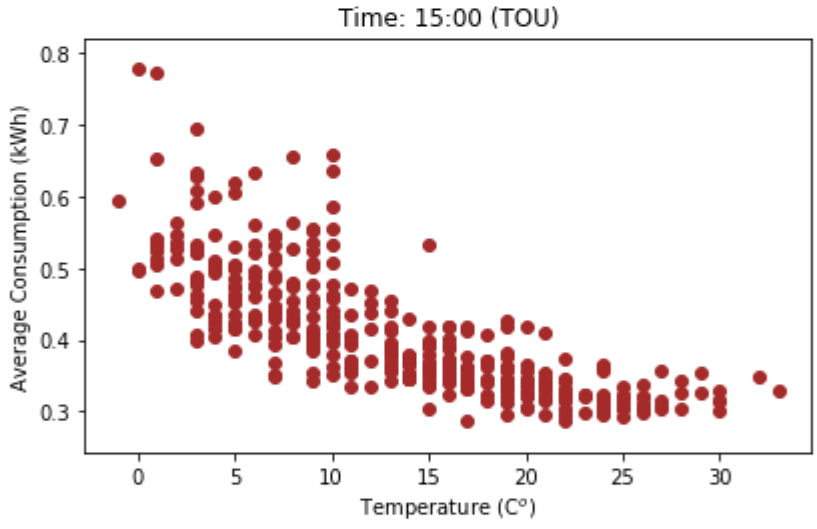

Time: 13:00 (non-TOU)
Average Consumption (kWh)
Temperature (C°)

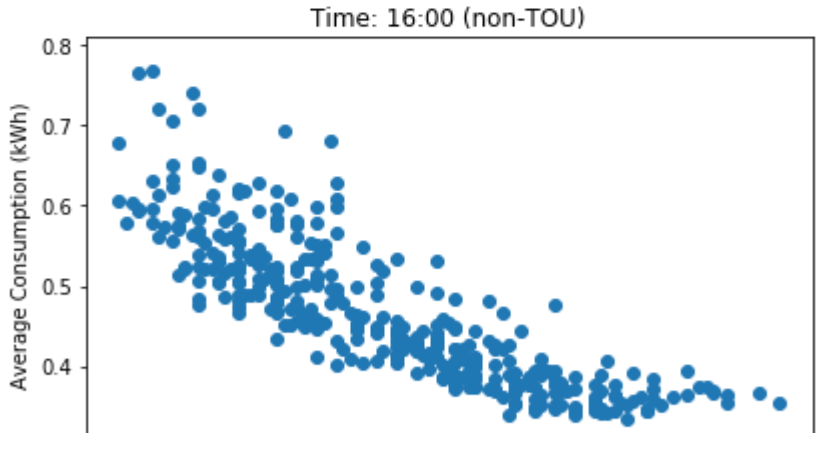

Time: 13:00 (TOU)
Average Consumption (kWh)
Temperature (C°)

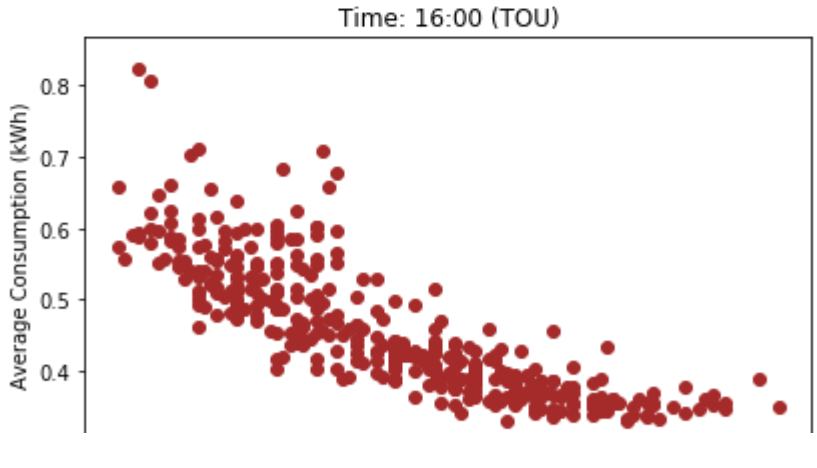

Time: 14:00 (non-TOU)
Average Consumption (kWh)
Temperature (C°)

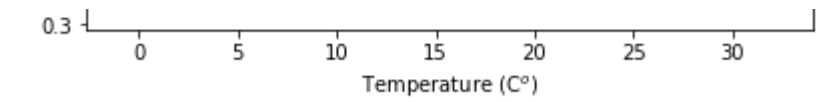

Time: 14:00 (TOU)
Average Consumption (kWh)
Temperature (C°)

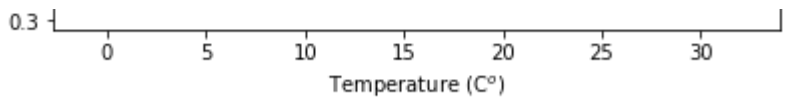

Time: 15:00 (non-TOU)
Average Consumption (kWh)
Temperature (C°)

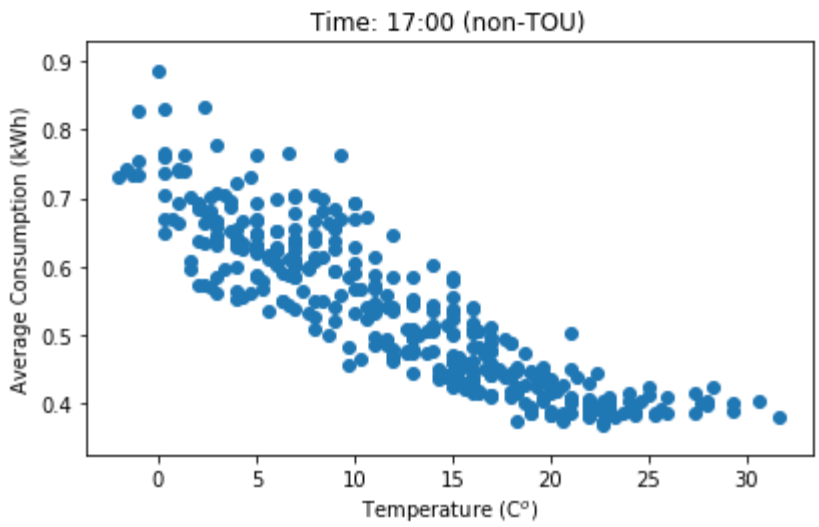

Time: 15:00 (TOU)
Average Consumption (kWh)
Temperature (C°)

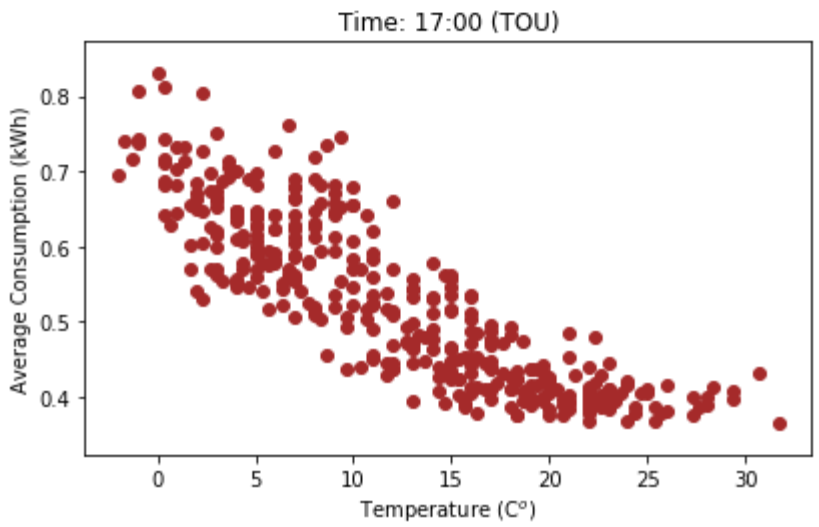

Time: 16:00 (non-TOU)
Average Consumption (kWh)

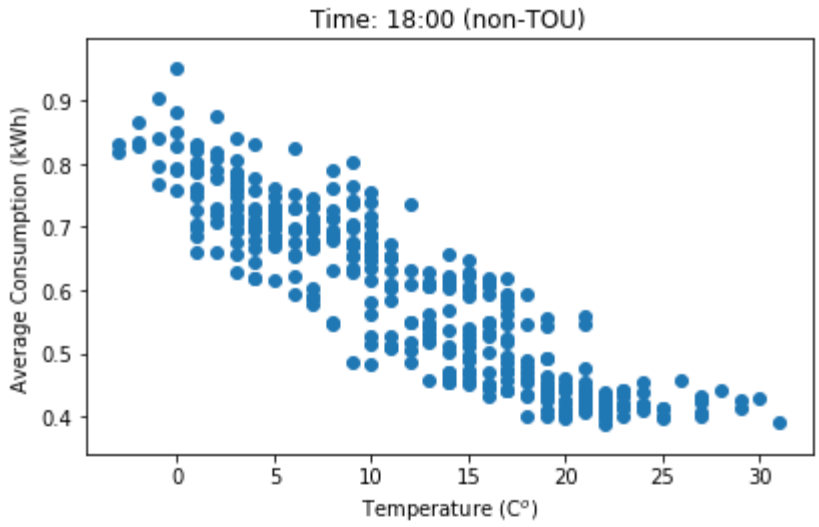

Time: 16:00 (TOU)
Average Consumption (kWh)

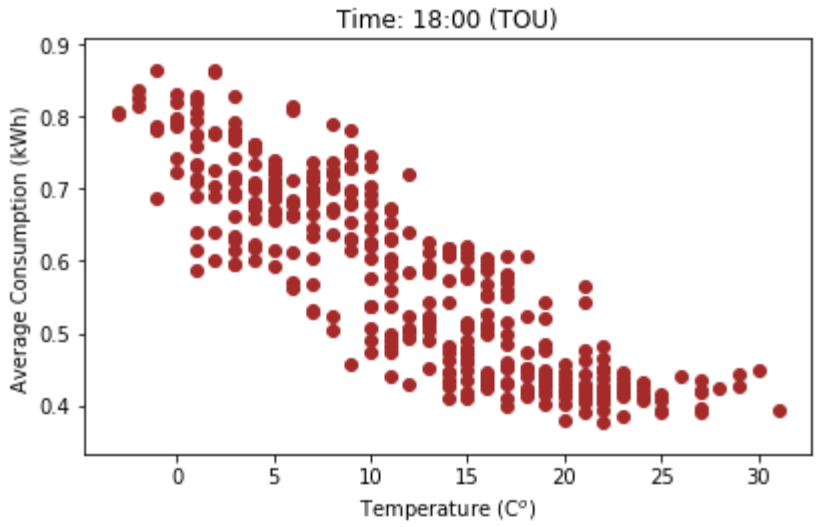

0.3
0
5
10
15
20
25
30
Temperature (C°)

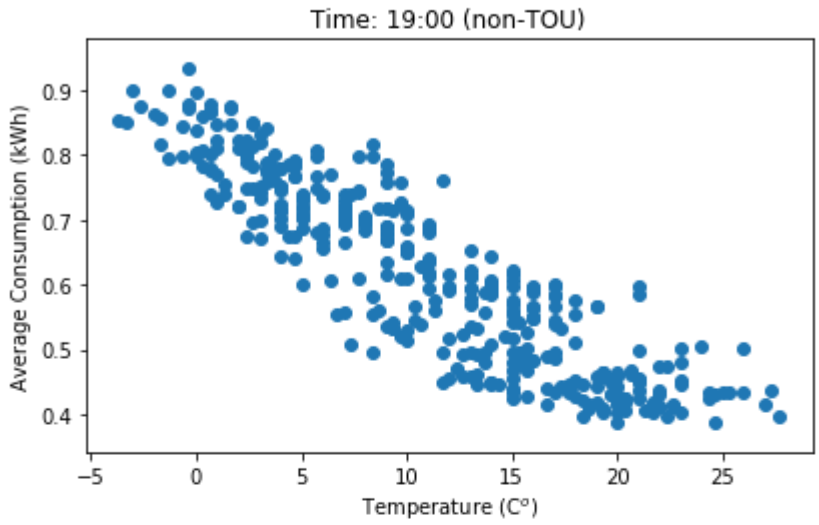

0.3
0
5
10
15
20
25
30
Temperature (C°)

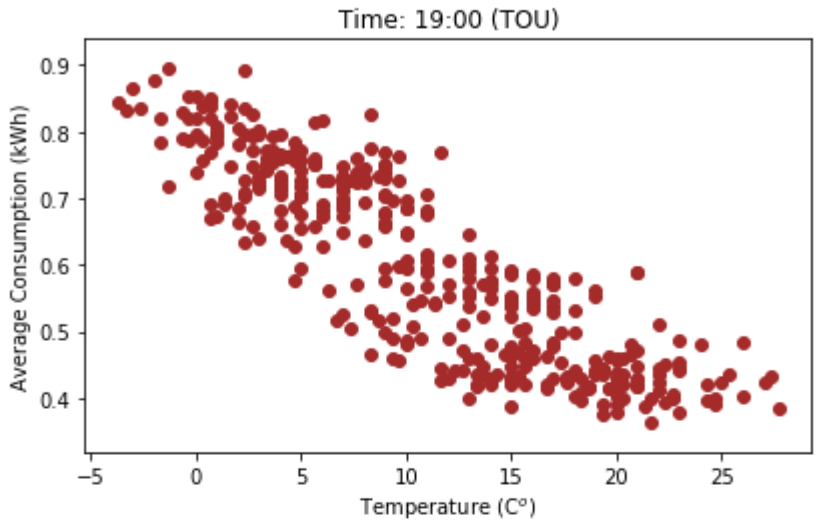

Time: 17:00 (non-TOU)
Average Consumption (kWh)
0.9
0.8
0.7
0.6
0.5
0.4
0
5
10
15
20
25
30
Temperature (C°)

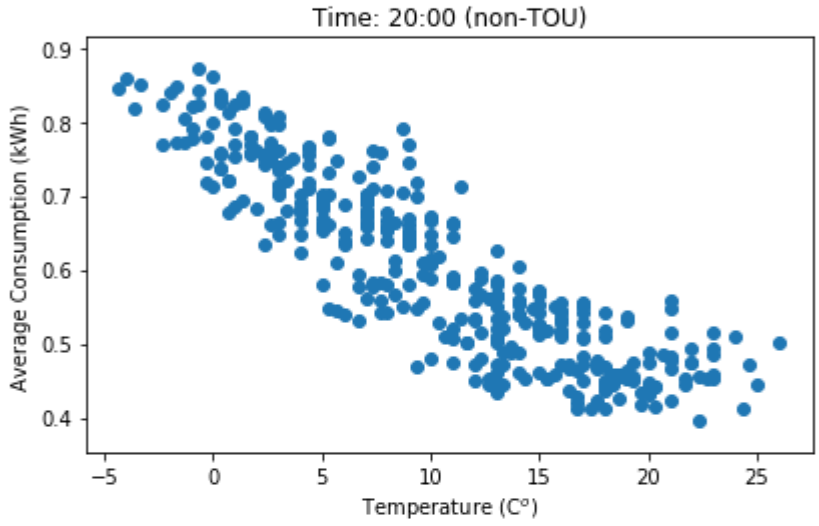

Time: 17:00 (TOU)
Average Consumption (kWh)
0.8
0.7
0.6
0.5
0.4
0
5
10
15
20
25
30
Temperature (C°)

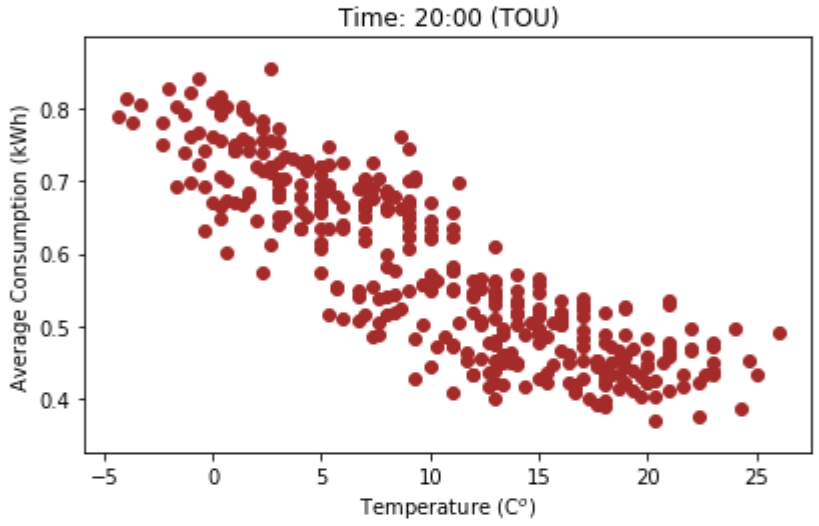

Time: 18:00 (non-TOU)
Average Consumption (kWh)
0.9
0.8
0.7
0.6
0.5
0.4
0
5
10
15
20
25
30
Temperature (C°)

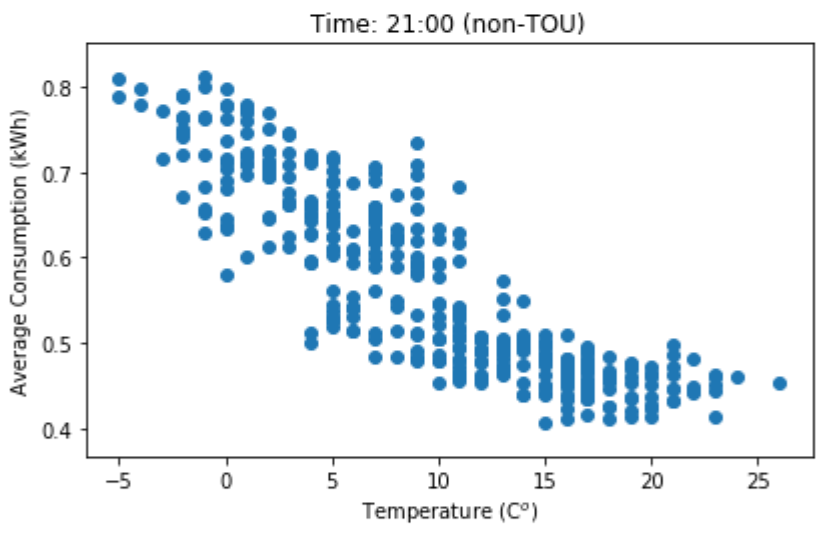

Time: 18:00 (TOU)
Average Consumption (kWh)
0.9
0.8
0.7
0.6
0.5
0.4
0
5
10
15
20
25
30
Temperature (C°)

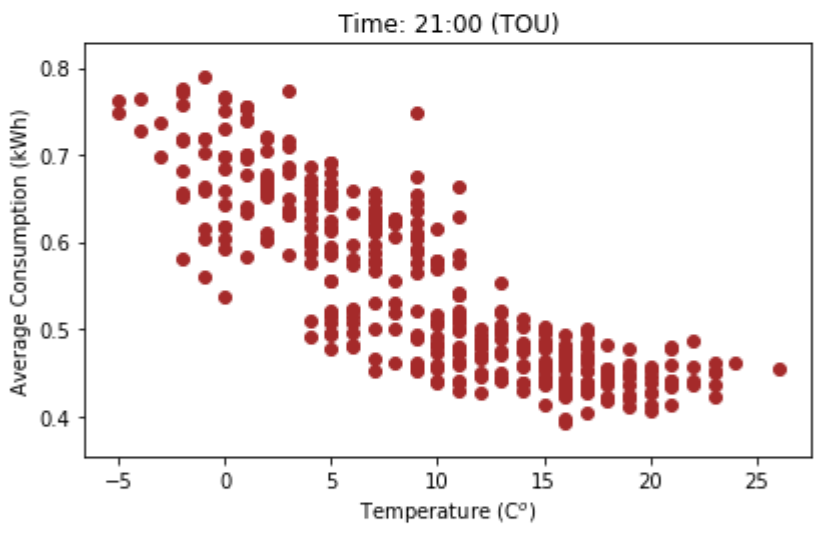

Time: 19:00 (non-TOU)
Average Consumption (kWh)
0.9
0.8
0.7
0.6
0.5
0.4
−5
0
5
10
15
20
25
Temperature (C°)


Time: 19:00 (TOU)
Average Consumption (kWh)
0.9
0.8
0.7
0.6
0.5
0.4
−5
0
5
10
15
20
25
Temperature (C°)


Time: 20:00 (non-TOU)
Average Consumption (kWh)
0.9
0.8
0.7
0.6
0.5
0.4
−5
0
5
10
15
20
25
Temperature (C°)

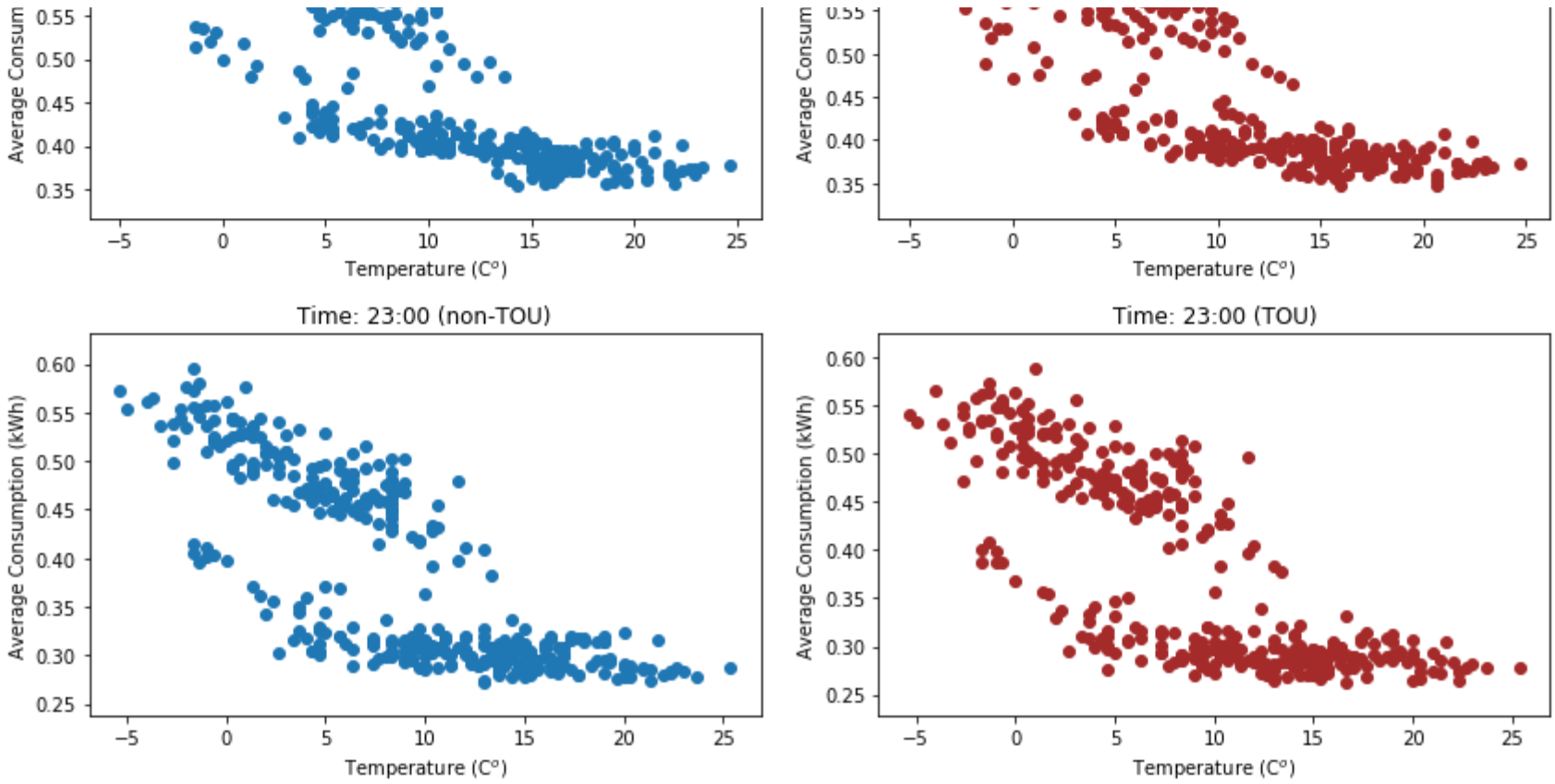

Time: 20:00 (TOU)
Average Consumption (kWh)
0.8
0.7
0.6
0.5
0.4
−5
0
5
10
15
20
25
Temperature (C°)


Time: 21:00 (non-TOU)
Average Consumption (kWh)
0.8
0.7
0.6
0.5
0.4
−5
0
5
10
15
20
25
Temperature (C°)


Time: 21:00 (TOU)
Average Consumption (kWh)
0.8
0.7
0.6
0.5
0.4
−5
0
5
10
15
20
25
Temperature (C°)

Time: 22:00 (non-TOU)
0.75
0.70
0.65
0.60

Time: 22:00 (TOU)
0.75
0.70
0.65
0.60



In [13]:
```
# # Baisc one without creating axes lists, used for future reference.
# for i in range(24):
#     ax = fig_all.add_subplot(24, 2, 2 * i + 1)
#     if i <= 9:
#         ax.scatter(df_wea1h.TempC[df_wea1h.GMT.str.contains('0' + str(i) + ':00:00')], df_Ntou1h.Average[df
_Ntou1h.GMT.str.contains('0' + str(i) + ':00:00')] )
#         ax.set_title('Time: 0' + str(i) + ':00 (non-TOU)')
#         ax.set_xlabel(r'Temperature (C$^o$)')
#         ax.set_ylabel('Average Consumption (kWh)')
#     else:
#         ax.scatter(df_wea1h.TempC[df_wea1h.GMT.str.contains(str(i) + ':00:00')], df_Ntou1h.Average[df_Ntou1
h.GMT.str.contains(str(i) + ':00:00')] )
#         ax.set_title('Time: ' + str(i) + ':00 (non-TOU)')
#         ax.set_xlabel(r'Temperature (C$^o$)')
#         ax.set_ylabel('Average Consumption (kWh)')
# for i in range(24):
#     ax = fig_all.add_subplot(24, 2, 2 * i + 2)
#     if i <= 9:
#         ax.scatter(df_wea1h.TempC[df_wea1h.GMT.str.contains('0' + str(i) + ':00:00')], df_tou1h.Average[df_
tou1h.GMT.str.contains('0' + str(i) + ':00:00')],  c = 'brown' )
#         ax.set_title('Time: 0' + str(i) + ':00 (TOU)')
#         ax.set_xlabel(r'Temperature (C$^o$)')
#         ax.set_ylabel('Average Consumption (kWh)')
#     else:
#         ax.scatter(df_wea1h.TempC[df_wea1h.GMT.str.contains(str(i) + ':00:00')], df_tou1h.Average[df_tou1h.
GMT.str.contains(str(i) + ':00:00')], c = 'brown' )
#         ax.set_title('Time: ' + str(i) + ':00 (TOU)')
#         ax.set_xlabel(r'Temperature (C$^o$)')
#         ax.set_ylabel('Average Consumption (kWh)')
# plt.tight_layout()
```

The correlation of average consumption and temperature is given below

In [14]:
```
def highlight_max(s):
    '''
    highlight the maximum in a Series yellow.
    '''
    is_max = s == s.max()
    return ['background-color: yellow' if v else '' for v in is_max]
```

In [15]:
```
# Let's first add correlation of average consumption and temperature on each plot
corr_nTou = []
corr_Tou = []
time_list = []
for i in range(24):
    if i <= 9:
        corr_nTou.append(np.corrcoef(df_wea1h.TempC[df_wea1h.GMT.str.contains('0' + str(i) + ':00:00')], df_N
tou1h.Average[df_Ntou1h.GMT.str.contains('0' + str(i) + ':00:00')])[0][1])
        corr_Tou.append(np.corrcoef(df_wea1h.TempC[df_wea1h.GMT.str.contains('0' + str(i) + ':00:00')], df_to
u1h.Average[df_tou1h.GMT.str.contains('0' + str(i) + ':00:00')])[0][1])
        time_list.append('0' + str(i) + ':00:00')
    else:
        corr_nTou.append(np.corrcoef(df_wea1h.TempC[df_wea1h.GMT.str.contains(str(i) + ':00:00')], df_Ntou1h.
Average[df_Ntou1h.GMT.str.contains(str(i) + ':00:00')])[0][1])
        corr_Tou.append(np.corrcoef(df_wea1h.TempC[df_wea1h.GMT.str.contains(str(i) + ':00:00')], df_tou1h.Av
erage[df_tou1h.GMT.str.contains(str(i) + ':00:00')])[0][1])
        time_list.append(str(i) + ':00:00')
corr_dict = {}
corr_dict['Time'] = time_list
corr_dict['Non-TOU'] = corr_nTou
corr_dict['TOU'] = corr_Tou
df_corr = pd.DataFrame(corr_dict)
print("The correlation between average consumption and temperature ")
df_corr.style.apply(highlight_max)
```

```
The correlation between average consumption and temperature
```

Out[15]:

	Time	Non-TOU	TOU
0	00:00:00	-0.750935	-0.719511
1	01:00:00	-0.742681	-0.708529
2	02:00:00	-0.749275	-0.701912
3	03:00:00	-0.768477	-0.733397
4	04:00:00	-0.805196	-0.779261
5	05:00:00	-0.686257	-0.618888
6	06:00:00	-0.326638	-0.236381
7	07:00:00	-0.607173	-0.586979
8	08:00:00	-0.856065	-0.842754
9	09:00:00	-0.896261	-0.863859
10	10:00:00	-0.847338	-0.813987
11	11:00:00	-0.791687	-0.761099
12	12:00:00	-0.768106	-0.744399
13	13:00:00	-0.783342	-0.76196
14	14:00:00	-0.784074	-0.759483
15	15:00:00	-0.783964	-0.7674
16	16:00:00	-0.846154	-0.828791
17	17:00:00	-0.886838	-0.860033
18	18:00:00	-0.899877	-0.866192
19	19:00:00	-0.896384	-0.868341
20	20:00:00	-0.888443	-0.86008
21	21:00:00	-0.871713	-0.839797
22	22:00:00	-0.838443	-0.817096
23	23:00:00	-0.790355	-0.770889

Fix the temperature and calculate mean of the data to see if there are difference between TOU and non-TOU case

In [6]:
```
# Combine the temperature dataframe and non-TOU and TOU data together, to construct 2 new combined dataframe
df_Ntou1h['TempC'] = df_wea1h['TempC'].round()
df_tou1h['TempC'] = df_wea1h['TempC'].round()
df_Ntou1h.groupby('TempC')['Total'].mean()
```

Out[6]:
```
TempC
-8.0     2422.617000
-7.0     2194.779500
-6.0     1859.903200
-5.0     1922.399167
-4.0     1813.153615
-3.0     1755.657652
-2.0     1853.209401
-1.0     1829.753208
 0.0     2043.474169
 1.0     2165.431803
 2.0     2070.932192
 3.0     1974.060900
 4.0     1906.644273
 5.0     1866.607695
 6.0     1782.732955
 7.0     1798.756414
 8.0     1761.014519
 9.0     1705.896461
 10.0    1681.594872
 11.0    1598.153580
 12.0    1464.691083
 13.0    1506.845987
 14.0    1492.618108
 15.0    1504.134397
 16.0    1480.637633
 17.0    1471.391500
 18.0    1486.999506
 19.0    1477.559386
 20.0    1436.509478
 21.0    1431.052278
 22.0    1422.638348
 23.0    1400.771698
 24.0    1422.030135
 25.0    1479.400702
 26.0    1402.175564
 27.0    1494.972879
 28.0    1512.884583
 29.0    1473.826500
 30.0    1395.854500
 31.0    1617.117667
 32.0    1424.387750
 33.0    1410.085500
Name: Total, dtype: float64
```

In [17]:
```
df_NdVSt = df_Ntou1h[df_Ntou1h.GMT.str.contains('0' + str(2) + ':00:00')]
df_NdVSt = pd.DataFrame(df_NdVSt.groupby('TempC')['Total'].mean() / (df_NdVSt.shape[1] - 4))
df_NdVSt = df_NdVSt.reset_index()
```

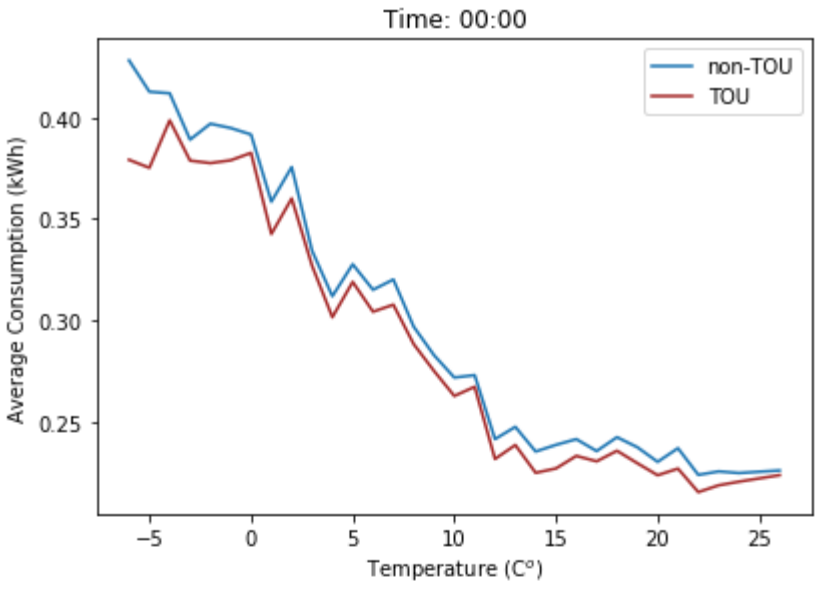
In [12]:
```
df_Ntou1h.head()
```

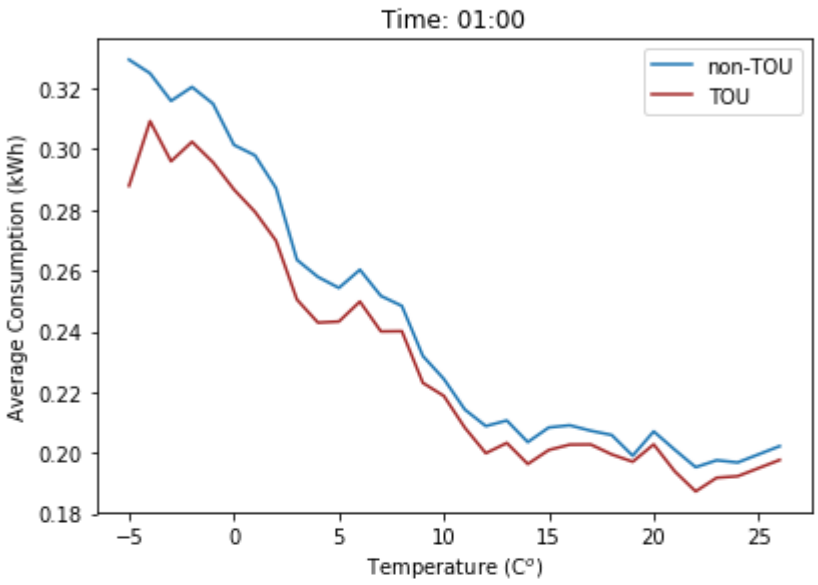
Out[12]:

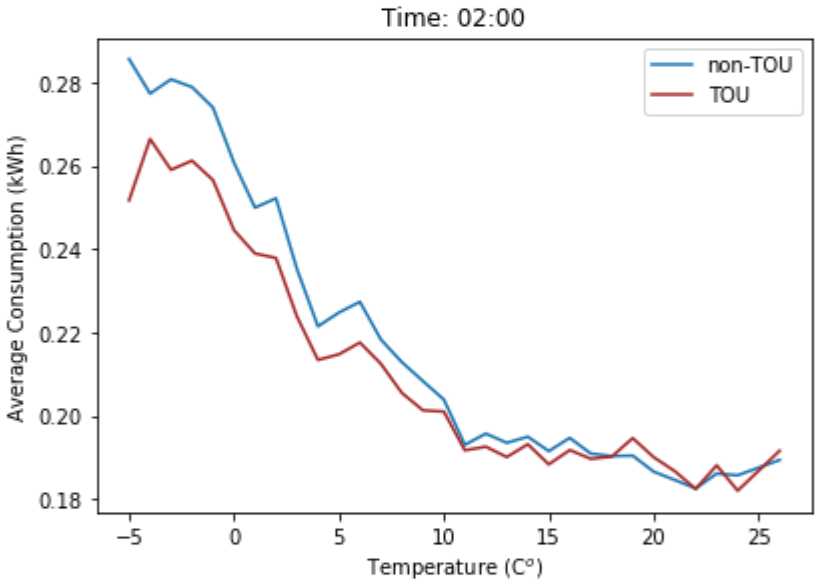
	GMT	N0000	N0001	N0002	N0003	N0004	N0005	N0006	N0007	N0008	...	N4166	N4167	N4168	N4169	N4170	N4171	N4172
0	2013-01-01 00:00:00	1.038	0.232	0.052	0.256	0.181	0.773	0.0	0.163	0.261	...	0.020	0.586	0.201	0.069	0.254	0.102	0.132
1	2013-01-01 01:00:00	0.484	0.267	0.101	0.192	0.185	0.359	0.0	0.152	0.090	...	0.115	0.464	0.194	0.070	0.163	0.176	0.140
2	2013-01-01 02:00:00	0.449	0.266	0.092	0.200	0.484	0.150	0.0	0.170	0.156	...	0.235	0.190	0.241	0.146	0.097	0.155	0.142
3	2013-01-01 03:00:00	0.390	0.230	0.034	0.262	0.132	0.155	0.0	0.152	0.094	...	0.021	0.157	0.185	0.232	0.137	0.095	0.147
4	2013-01-01 04:00:00	0.936	0.268	0.052	0.205	0.183	0.163	0.0	0.165	0.155	...	0.021	0.175	0.312	0.078	0.089	0.094	0.161

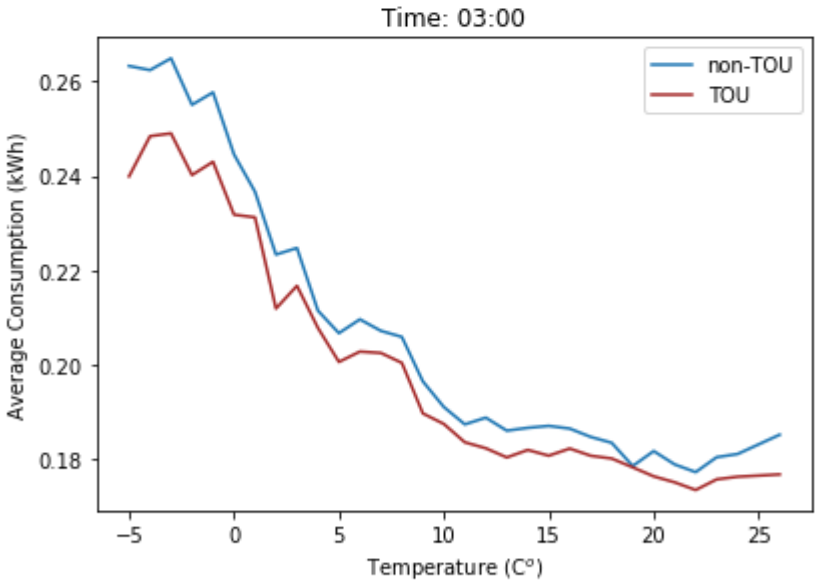
5 rows × 4177 columns

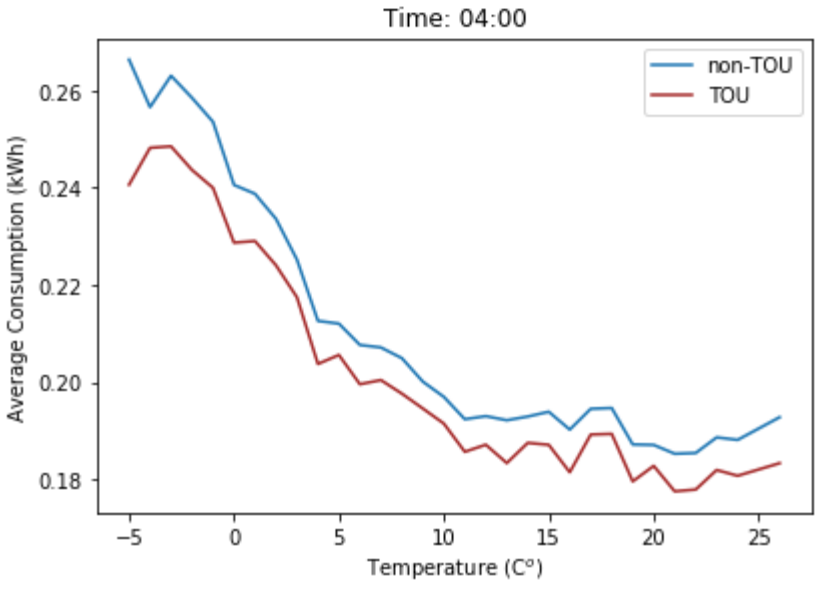
In [18]:
```
# non-TOU vs TOU in terms of average (over both users and the time) hourly energy consumption v.s. temperatur
e
fig_all = plt.figure(figsize = (6,100))
ax_Ntou = [] # store subplot objects
ax_tou = [] # store subplot objects
for i in range(24):
    ax_Ntou.append(fig_all.add_subplot(24, 1, i+1))
    if i <= 9:
        # calculate the average consumption grouped by different temperatures
        df_NdVSt = df_Ntou1h[df_Ntou1h.GMT.str.contains('0' + str(i) + ':00:00')] #non-tou demand vs temperat
ure
        df_NdVSt = df_NdVSt.groupby('TempC')['Total'].mean() / (df_NdVSt.shape[1] - 4)
        df_NdVSt = df_NdVSt.reset_index()
        ax_Ntou[-1].plot(df_NdVSt.TempC, df_NdVSt.Total, label = 'non-TOU' )
        df_dVSt = df_tou1h[df_tou1h.GMT.str.contains('0' + str(i) + ':00:00')]
        df_dVSt = df_dVSt.groupby('TempC')['Total'].mean() / (df_dVSt.shape[1] - 4)
        df_dVSt = df_dVSt.reset_index()
        ax_Ntou[-1].plot(df_dVSt.TempC, df_dVSt.Total,  c = 'brown', label = 'TOU' )
        ax_Ntou[-1].set_title('Time: 0' + str(i) + ':00')
        ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_Ntou[-1].set_ylabel('Average Consumption (kWh)')
        ax_Ntou[-1].legend()
    else:
        df_NdVSt = df_Ntou1h[df_Ntou1h.GMT.str.contains(str(i) + ':00:00')]
        df_NdVSt = df_NdVSt.groupby('TempC')['Total'].mean() / (df_NdVSt.shape[1] - 4)
        df_NdVSt = df_NdVSt.reset_index()
        ax_Ntou[-1].plot(df_NdVSt.TempC, df_NdVSt.Total, label = 'non-TOU')
        df_dVSt = df_tou1h[df_tou1h.GMT.str.contains(str(i) + ':00:00')]
        df_dVSt = df_dVSt.groupby('TempC')['Total'].mean() / (df_dVSt.shape[1] - 4)
        df_dVSt = df_dVSt.reset_index()
        ax_Ntou[-1].plot(df_dVSt.TempC, df_dVSt.Total,  c = 'brown', label = 'TOU' )
        ax_Ntou[-1].set_title('Time: ' + str(i) + ':00')
        ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_Ntou[-1].set_ylabel('Average Consumption (kWh)')
        ax_Ntou[-1].legend()
plt.tight_layout()
```

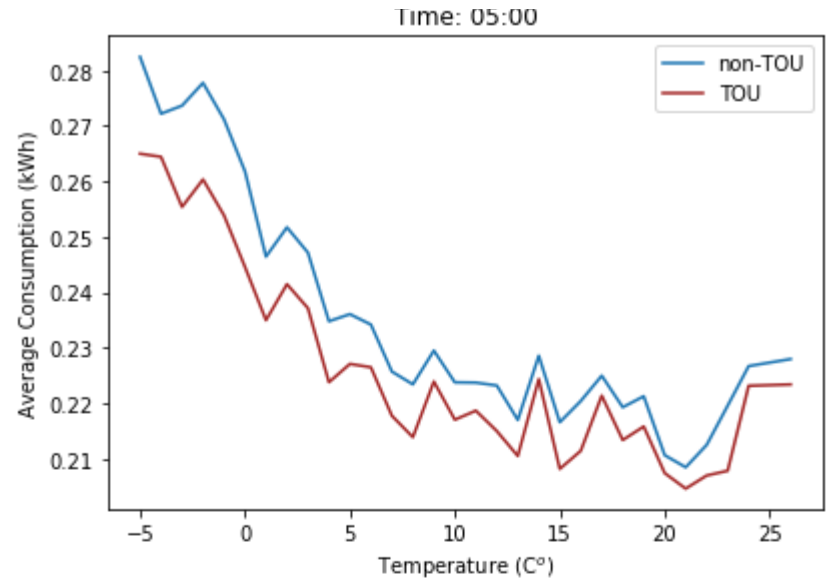

Time: 00:00
non-TOU
TOU
Average Consumption (kWh)
0.40
0.35
0.30
0.25
−5
0
5
10
15
20
25
Temperature (C°)

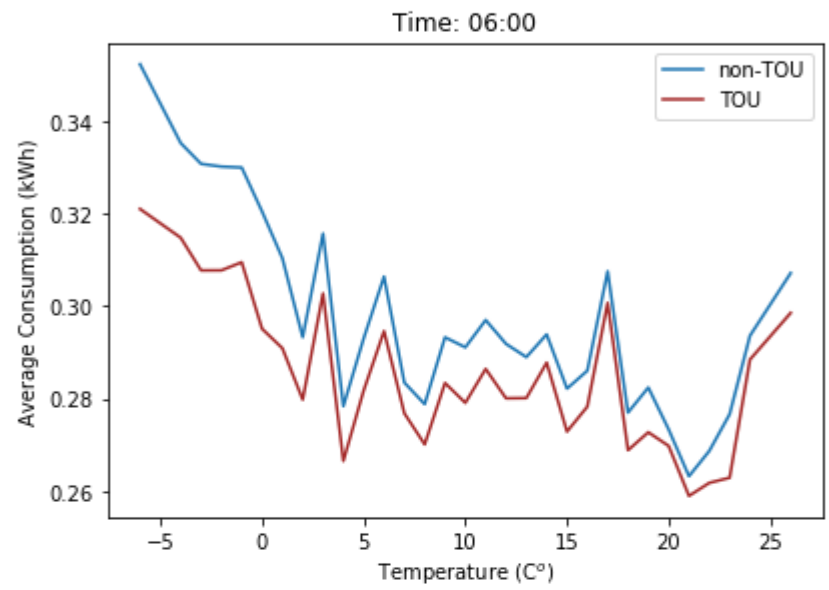

Time: 01:00
non-TOU
TOU
Average Consumption (kWh)
0.32
0.30
0.28
0.26
0.24
0.22
0.20
0.18
−5
0
5
10
15
20
25
Temperature (C°)

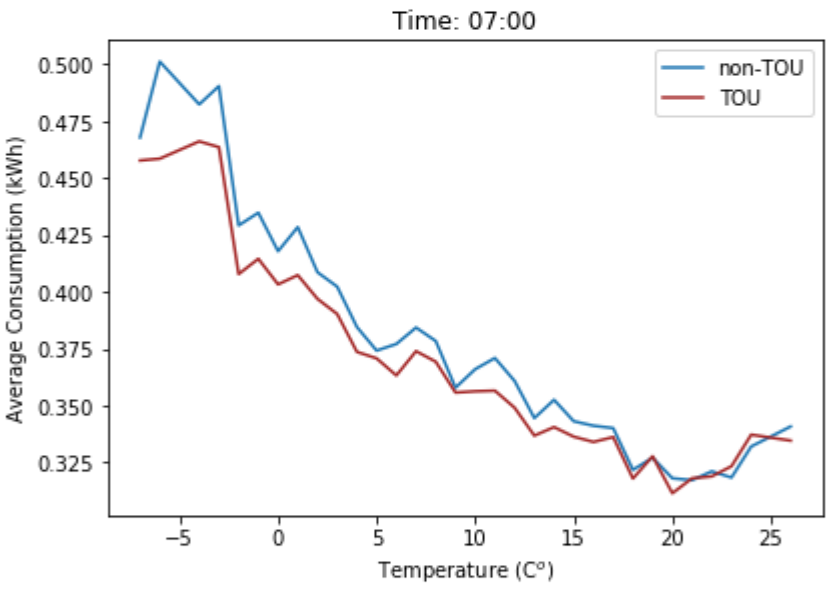

Time: 02:00
non-TOU
TOU
Average Consumption (kWh)
0.28
0.26
0.24
0.22
0.20
0.18
−5
0
5
10
15
20
25
Temperature (C°)

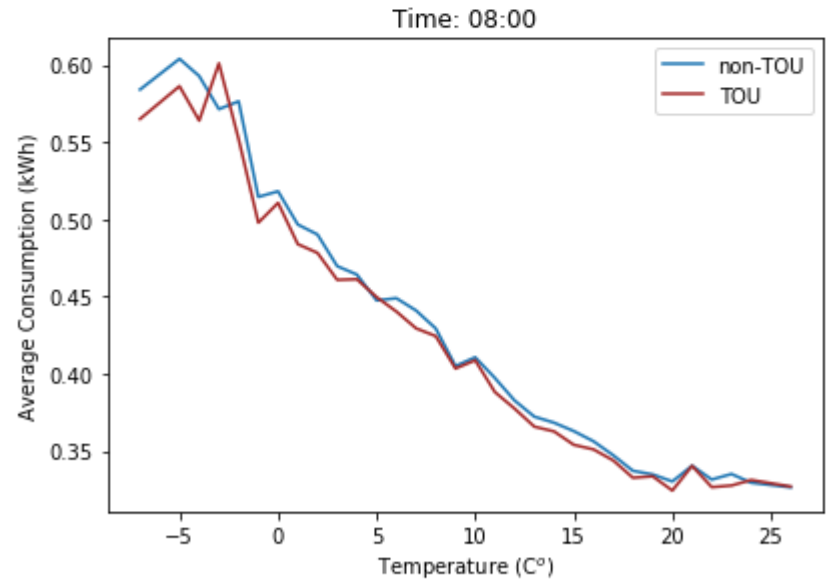

Time: 03:00
non-TOU
TOU
Average Consumption (kWh)
0.26
0.24
0.22
0.20
0.18
−5
0
5
10
15
20
25
Temperature (C°)

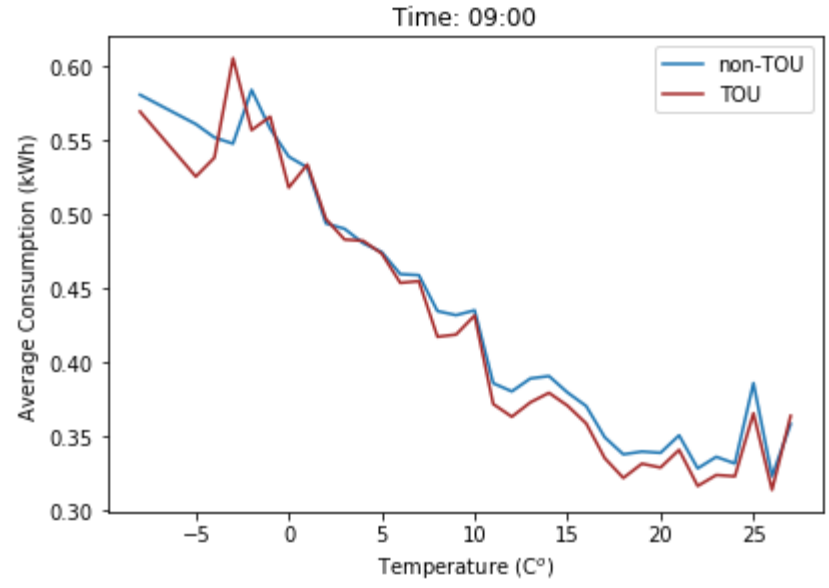

Time: 04:00
non-TOU
TOU
Average Consumption (kWh)
0.26
0.24
0.22
0.20
0.18
−5
0
5
10
15
20
25
Temperature (C°)


Time: 05:00
non-TOU
TOU
Average Consumption (kWh)
0.28
0.27
0.26
0.25
0.24
0.23
0.22
0.21
−5
0
5
10
15
20
25
Temperature (C°)

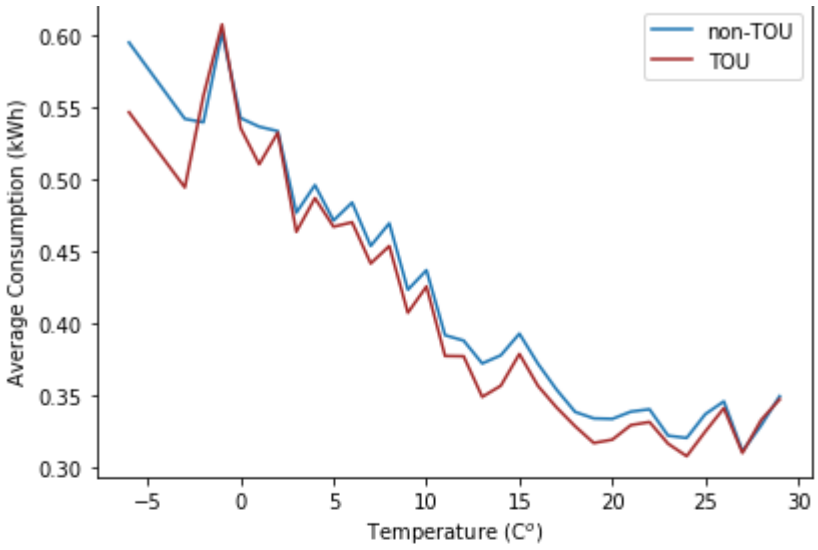

Time: 06:00
non-TOU
TOU
Average Consumption (kWh)
0.34
0.32
0.30
0.28
0.26
−5
0
5
10
15
20
25
Temperature (C°)


Time: 07:00
non-TOU
TOU
Average Consumption (kWh)
0.500
0.475
0.450
0.425
0.400
0.375
0.350
0.325
−5
0
5
10
15
20
25
Temperature (C°)

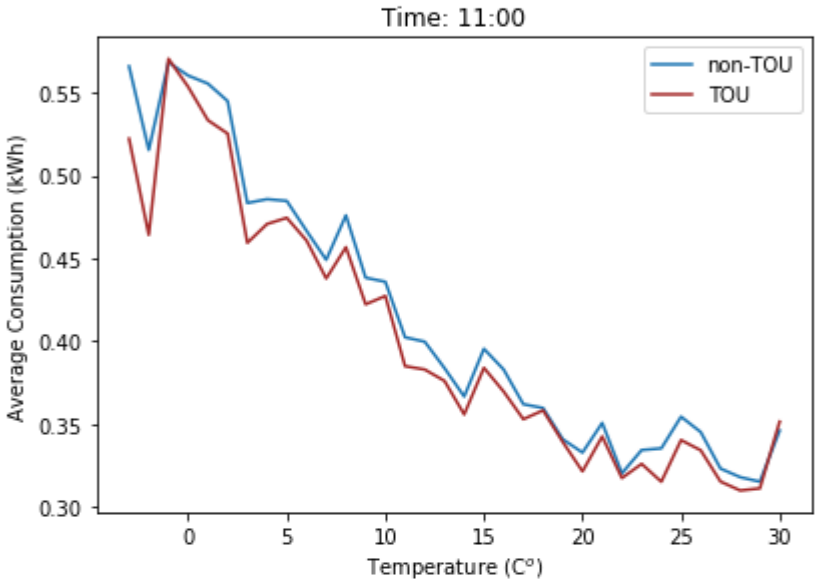

Time: 08:00
non-TOU
TOU
Average Consumption (kWh)
0.60
0.55
0.50
0.45
0.40
0.35
−5
0
5
10
15
20
25
Temperature (C°)


Time: 09:00
non-TOU
TOU
Average Consumption (kWh)
0.60
0.55
0.50
0.45
0.40
0.35
0.30
−5
0
5
10
15
20
25
Temperature (C°)

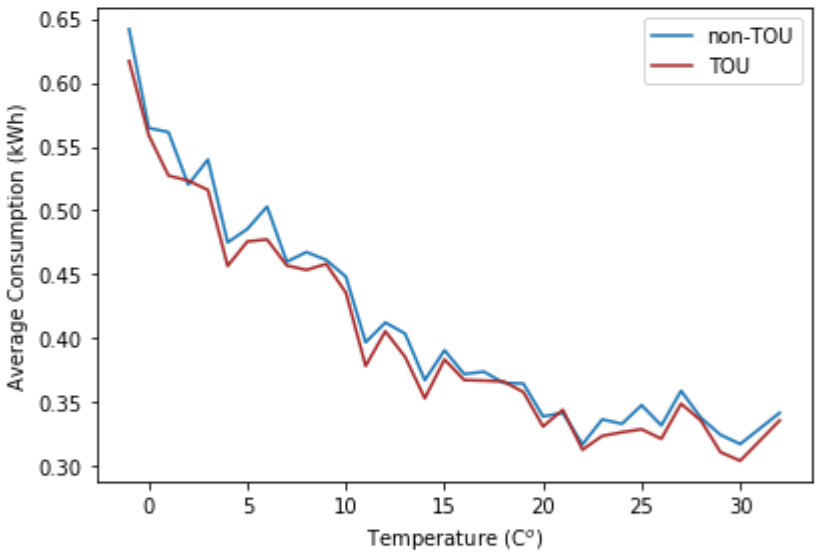
Time: 10:00



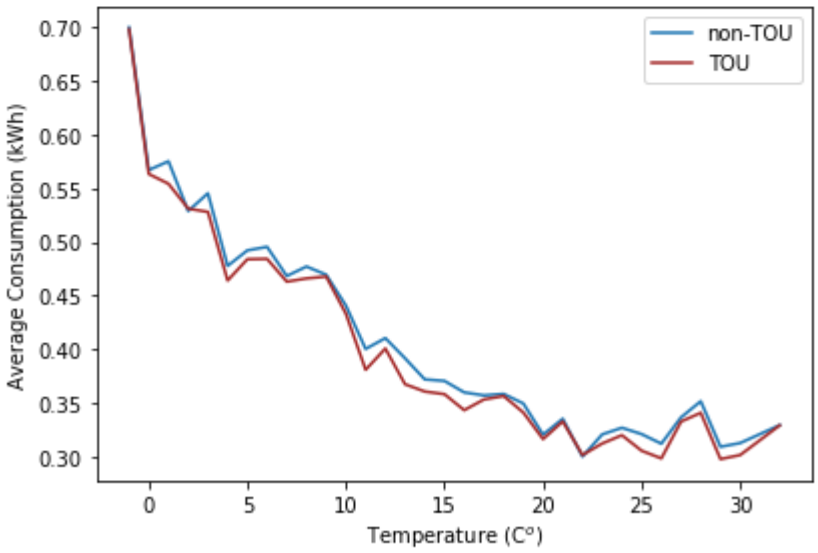
Time: 11:00



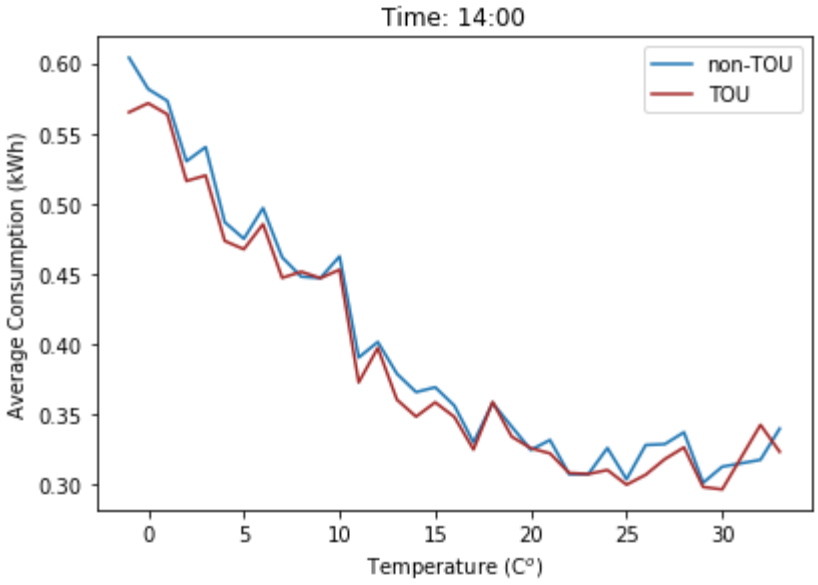
Time: 12:00



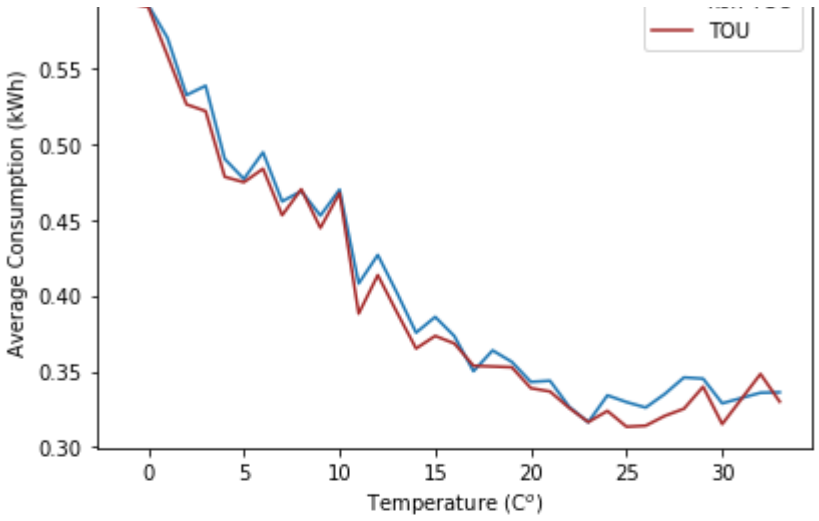
Time: 13:00

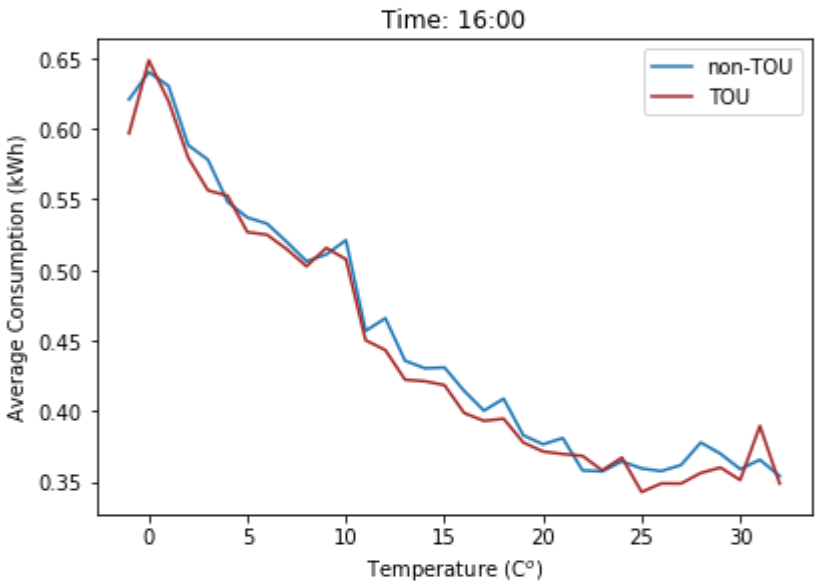


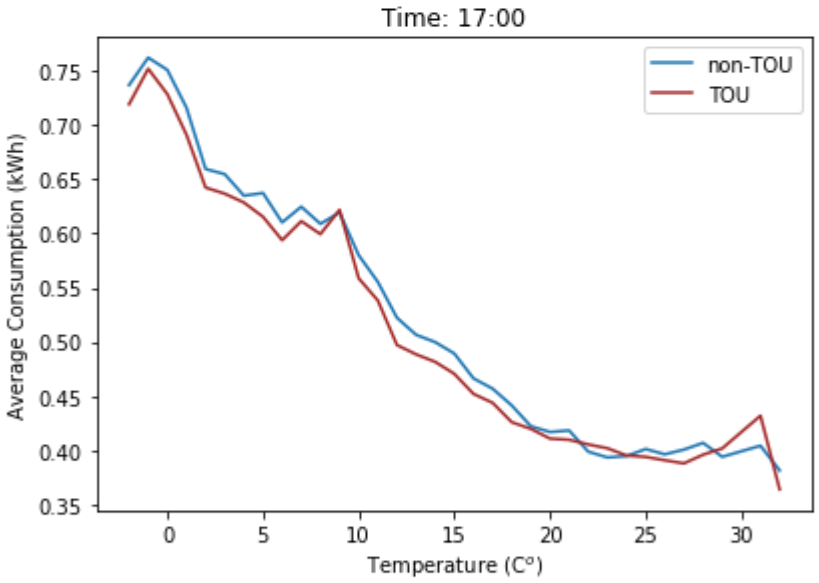
Time: 14:00

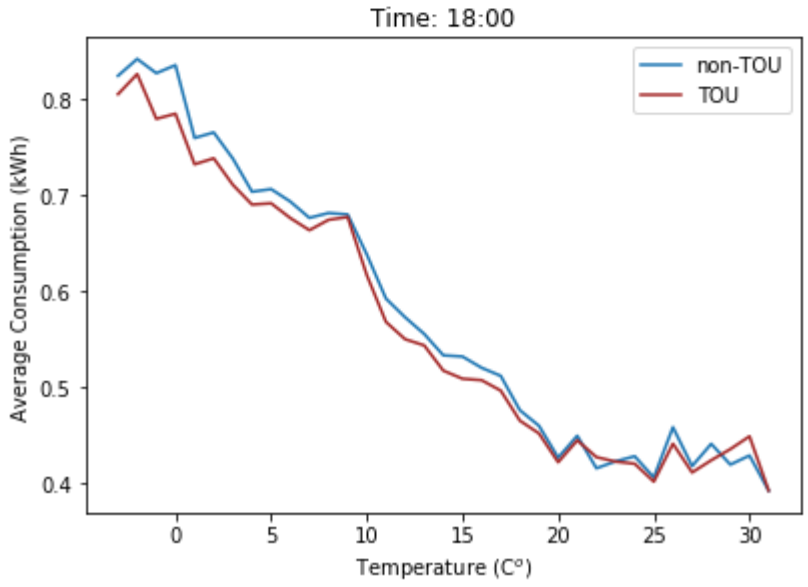


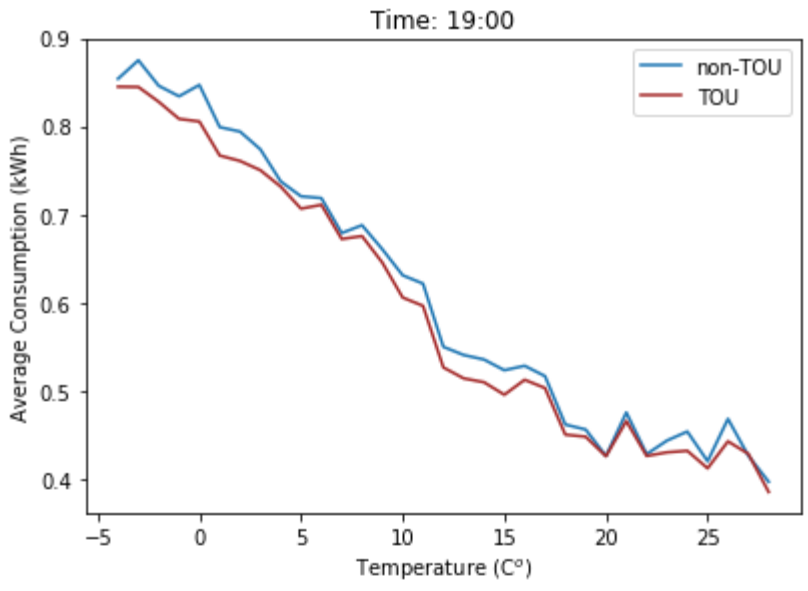
Time: 15:00


TOU
Average Consumption (kWh)
0.55
0.50
0.45
0.40
0.35
0.30
0
5
10
15
20
25
30
Temperature (C°)

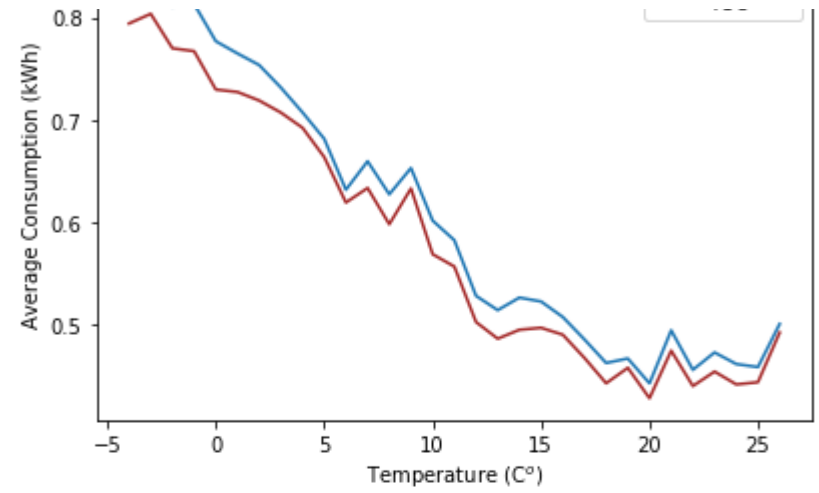

Time: 16:00
non-TOU
TOU
Average Consumption (kWh)
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0
5
10
15
20
25
30
Temperature (C°)

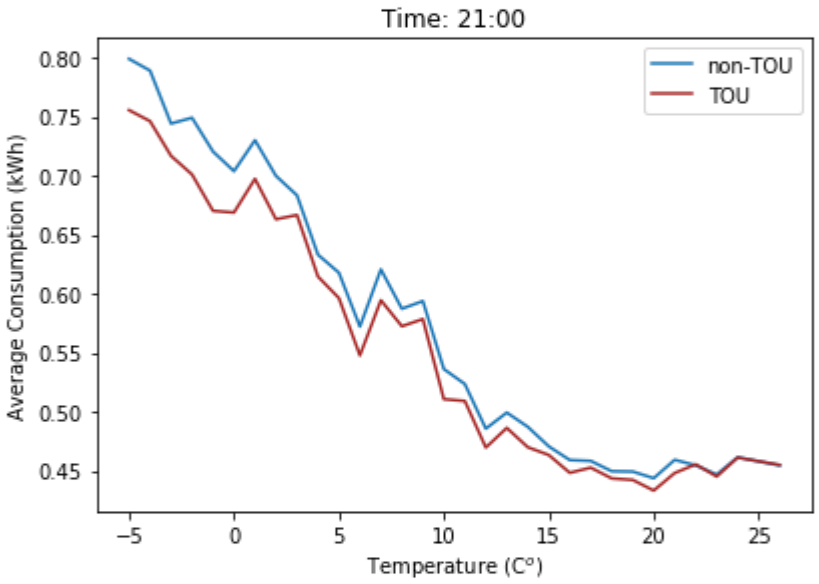

Time: 17:00
non-TOU
TOU
Average Consumption (kWh)
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0
5
10
15
20
25
30
Temperature (C°)

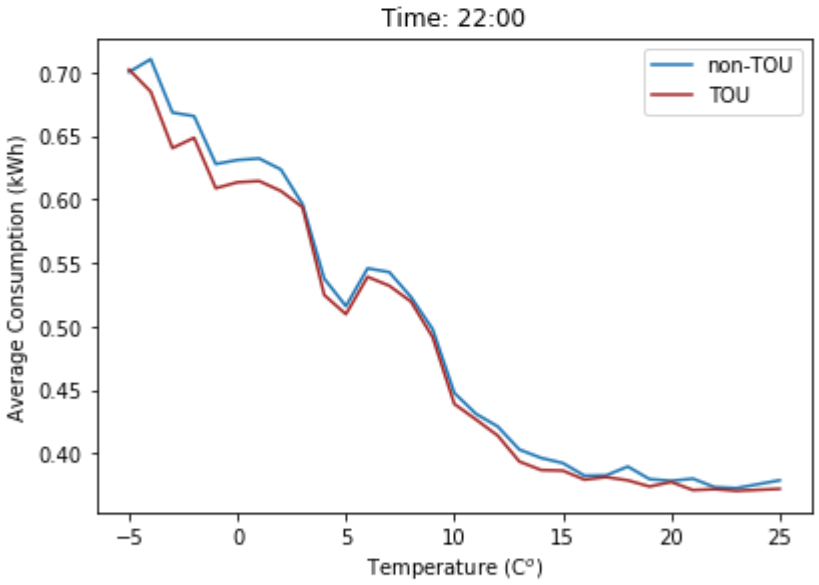

Time: 18:00
non-TOU
TOU
Average Consumption (kWh)
0.8
0.7
0.6
0.5
0.4
0
5
10
15
20
25
30
Temperature (C°)

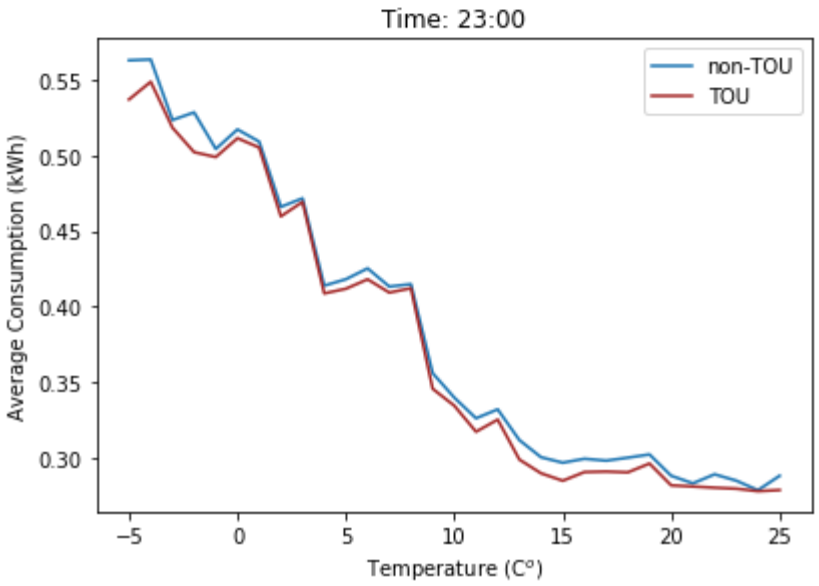

Time: 19:00
non-TOU
TOU
Average Consumption (kWh)
0.9
0.8
0.7
0.6
0.5
0.4
−5
0
5
10
15
20
25
Temperature (C°)

Time: 20:00
non-TOU
TOU



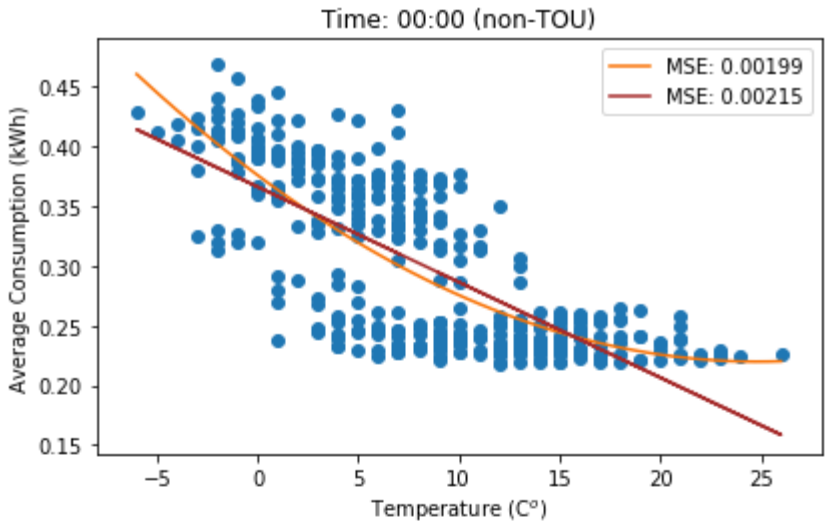


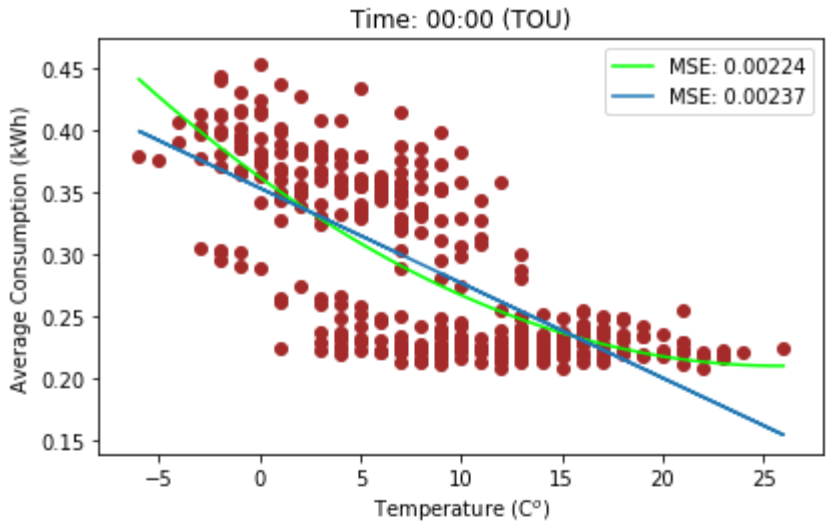


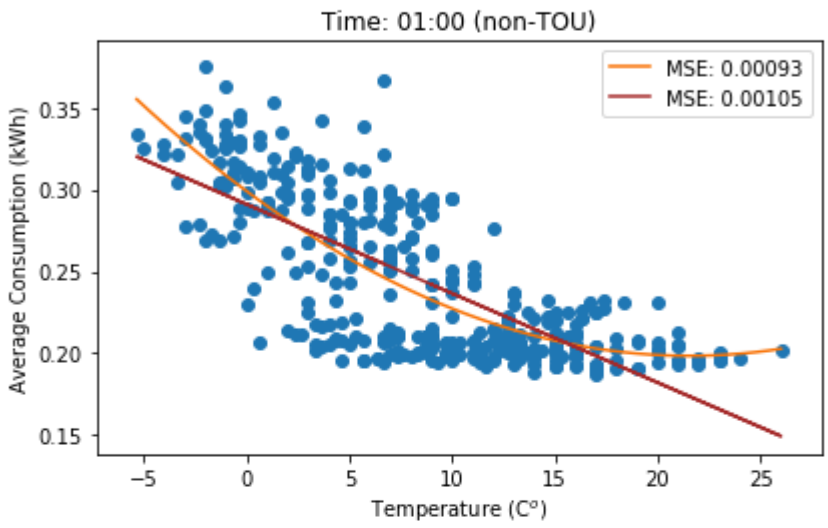


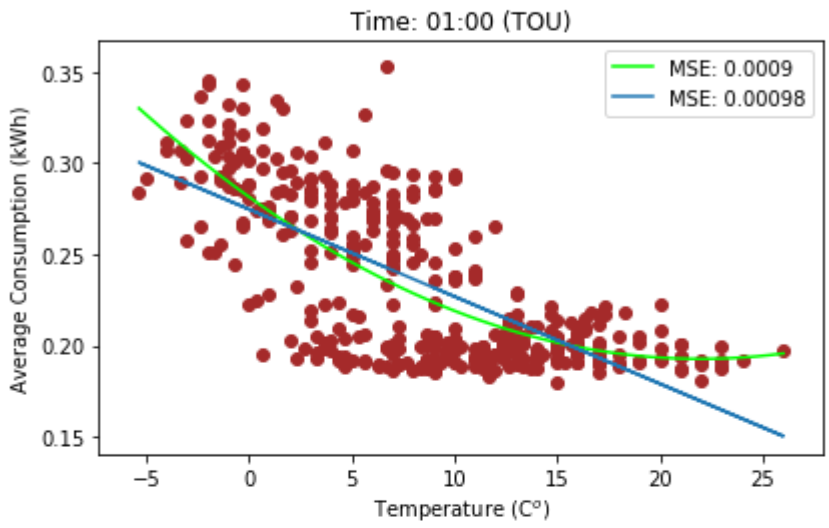
Add linear and polynomial regression, and see which results in a less MSE.

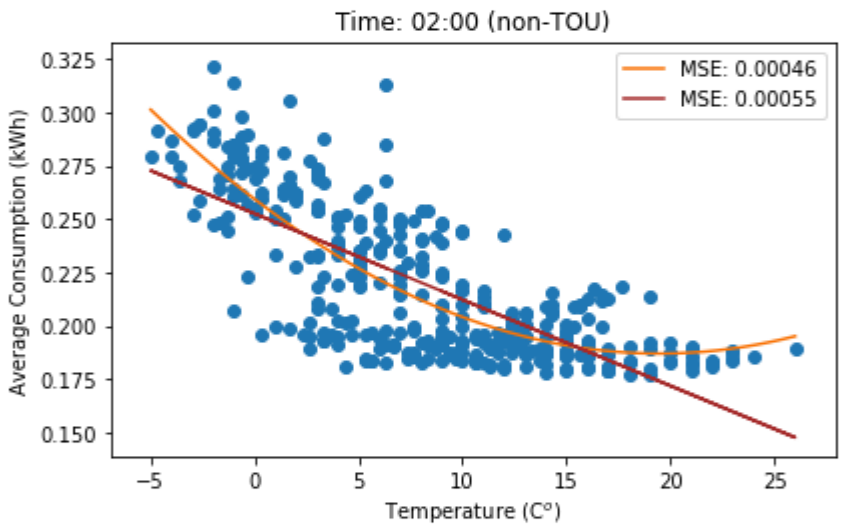
In [20]:
```
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
# non-TOU vs TOU in terms of showing the elements in Average column such that they are hourly energy consumpt
ion v.s. temperature over 365 days
fig_all = plt.figure(figsize = (12,90))
ax_Ntou = [] # store subplot objects
ax_tou = [] # store subplot objects
lm = LinearRegression()
for i in range(24):
    ax_Ntou.append(fig_all.add_subplot(24, 2, 2 * i + 1))
    if i <= 9:
        x = df_wea1h.TempC[df_wea1h.GMT.str.contains('0' + str(i) + ':00:00')].values
        y = df_Ntou1h.Average[df_Ntou1h.GMT.str.contains('0' + str(i) + ':00:00')].values
        xp = np.linspace(min(x), max(x), 50) # polynomial regression
        z = np.polyfit(x, y, 2)
        p = np.poly1d(z)
        ax_Ntou[-1].plot(xp, p(xp), color = 'xkcd:orange', label = 'MSE: ' + str(round(mean_squared_error(y,
p(x)), 5)))
        x = x.reshape(-1, 1)
        y = y.reshape(-1, 1)
        lm.fit(x, y)
        ax_Ntou[-1].plot(x, lm.predict(x), color = 'brown', label = 'MSE: ' + str(round(mean_squared_error(y,
lm.predict(x)), 5)))
        ax_Ntou[-1].scatter(x, y)
        ax_Ntou[-1].set_title('Time: 0' + str(i) + ':00 (non-TOU)')
        ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_Ntou[-1].set_ylabel('Average Consumption (kWh)')
        ax_Ntou[-1].legend()
    else:
        x = df_wea1h.TempC[df_wea1h.GMT.str.contains(str(i) + ':00:00')].values
        y = df_Ntou1h.Average[df_Ntou1h.GMT.str.contains(str(i) + ':00:00')].values
        xp = np.linspace(min(x), max(x), 50) # polynomial regression
        z = np.polyfit(x, y, 2)
        p = np.poly1d(z)
        ax_Ntou[-1].plot(xp, p(xp), color = 'xkcd:orange', label = 'MSE: ' + str(round(mean_squared_error(y,
p(x)), 5)))
        x = x.reshape(-1, 1)
        y = y.reshape(-1, 1)
        lm.fit(x, y)
        ax_Ntou[-1].plot(x, lm.predict(x), color = 'brown', label = 'MSE: ' + str(round(mean_squared_error(y,
lm.predict(x)), 5)))
        ax_Ntou[-1].scatter(x, y)
        ax_Ntou[-1].set_title('Time: ' + str(i) + ':00 (non-TOU)')
        ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_Ntou[-1].set_ylabel('Average Consumption (kWh)')
        ax_Ntou[-1].legend()
for i in range(24):
    ax_tou.append(fig_all.add_subplot(24, 2, 2 * i + 2))
    if i <= 9:
        x = df_wea1h.TempC[df_wea1h.GMT.str.contains('0' + str(i) + ':00:00')].values
        y = df_tou1h.Average[df_tou1h.GMT.str.contains('0' + str(i) + ':00:00')].values
        xp = np.linspace(min(x), max(x), 50) # polynomial regression
        z = np.polyfit(x, y, 2)
        p = np.poly1d(z)
        ax_tou[-1].plot(xp, p(xp), color = 'lime', label = 'MSE: ' + str(round(mean_squared_error(y, p(x)), 5
)))
        x = x.reshape(-1, 1)
        y = y.reshape(-1, 1)
        lm.fit(x, y)
        ax_tou[-1].plot(x, lm.predict(x), label = 'MSE: ' + str(round(mean_squared_error(y, lm.predict(x)), 5
)))
        ax_tou[-1].scatter(x, y, color = 'brown')
        ax_tou[-1].set_title('Time: 0' + str(i) + ':00 (TOU)')
        ax_tou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_tou[-1].set_ylabel('Average Consumption (kWh)')
        ax_tou[-1].legend()
    else:
        x = df_wea1h.TempC[df_wea1h.GMT.str.contains(str(i) + ':00:00')].values
        y = df_tou1h.Average[df_tou1h.GMT.str.contains(str(i) + ':00:00')].values
        xp = np.linspace(min(x), max(x), 50) # polynomial regression
        z = np.polyfit(x, y, 2)
        p = np.poly1d(z)
        ax_tou[-1].plot(xp, p(xp), color = 'lime', label = 'MSE: ' + str(round(mean_squared_error(y, p(x)), 5
)))
        x = x.reshape(-1, 1)
        y = y.reshape(-1, 1)
        lm.fit(x, y)
        ax_tou[-1].plot(x, lm.predict(x), label = 'MSE: ' + str(round(mean_squared_error(y, lm.predict(x)), 5
)))
        ax_tou[-1].scatter(x, y, color = 'brown')
        ax_tou[-1].set_title('Time: ' + str(i) + ':00 (TOU)')
        ax_tou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_tou[-1].set_ylabel('Average Consumption (kWh)')
        ax_tou[-1].legend()
plt.tight_layout()
```

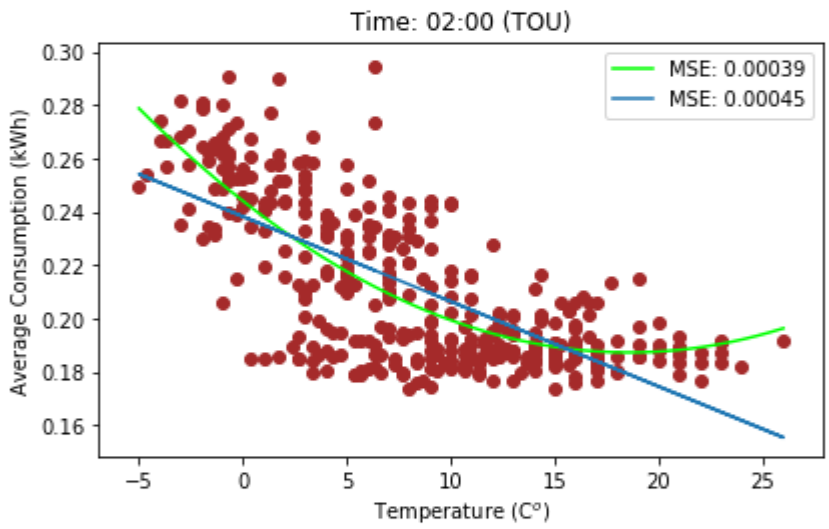

Time: 00:00 (non-TOU)
MSE: 0.00199
MSE: 0.00215
Average Consumption (kWh)
Temperature (C°)

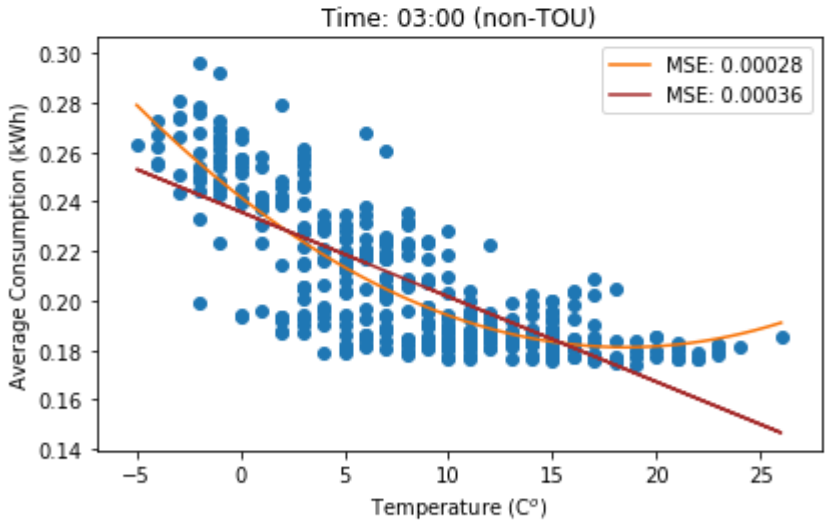

Time: 00:00 (TOU)
MSE: 0.00224
MSE: 0.00237
Average Consumption (kWh)
Temperature (C°)

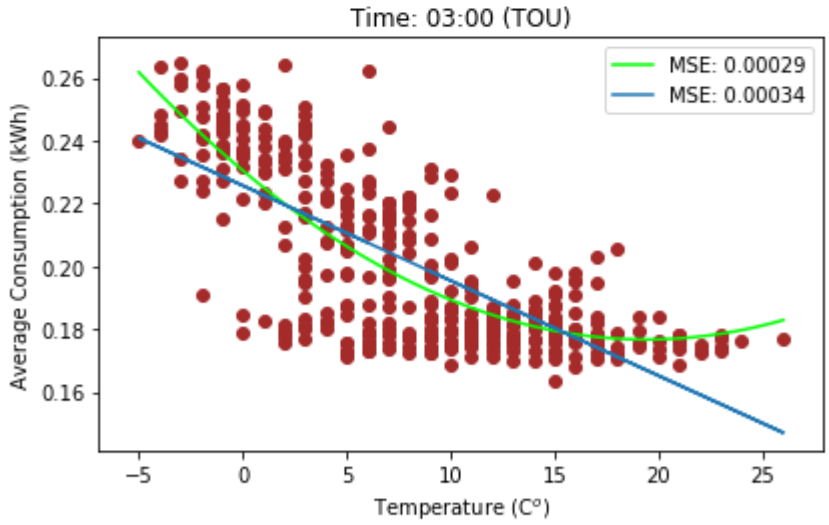

Time: 01:00 (non-TOU)
MSE: 0.00093
MSE: 0.00105
Average Consumption (kWh)
Temperature (C°)

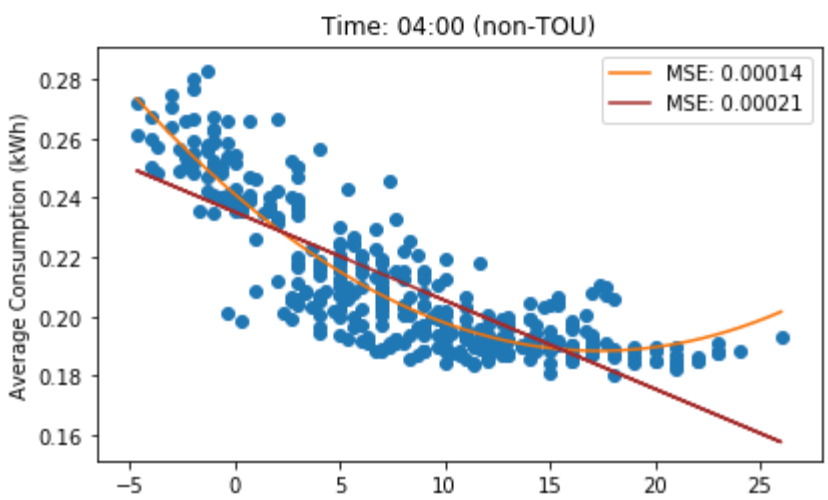

Time: 01:00 (TOU)
MSE: 0.0009
MSE: 0.00098
Average Consumption (kWh)
Temperature (C°)

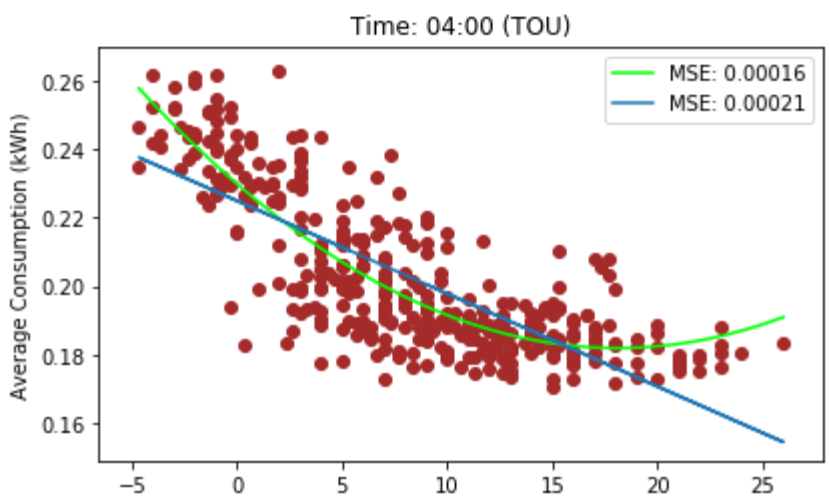

Time: 02:00 (non-TOU)
MSE: 0.00046
MSE: 0.00055
Average Consumption (kWh)
Temperature (C°)

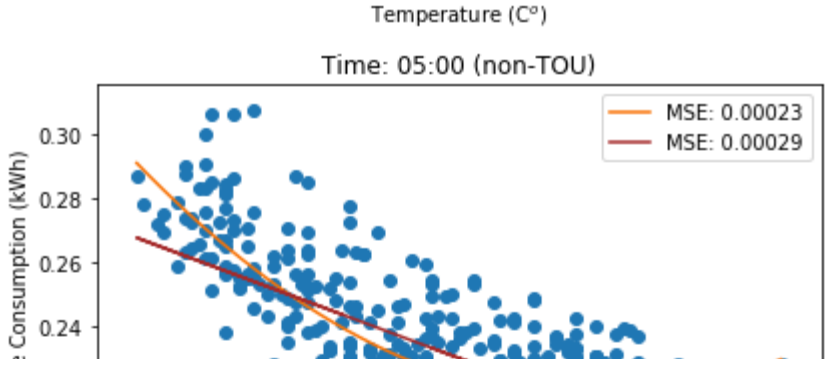

Time: 02:00 (TOU)
MSE: 0.00039
MSE: 0.00045
Average Consumption (kWh)
Temperature (C°)

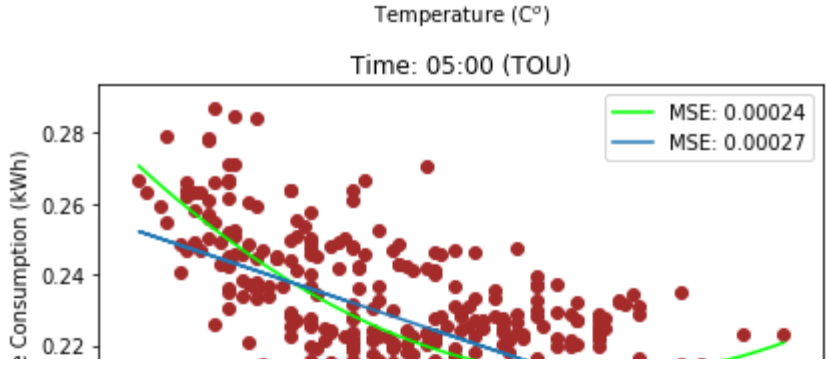

Time: 03:00 (non-TOU)
MSE: 0.00028
MSE: 0.00036
Average Consumption (kWh)
Temperature (C°)

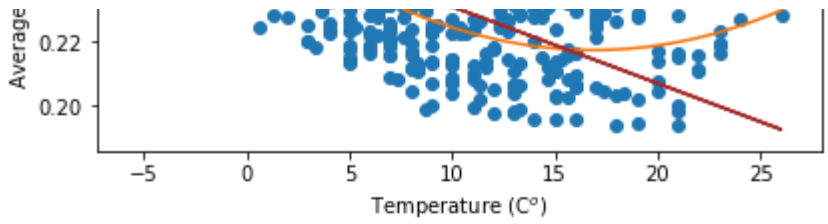

Time: 03:00 (TOU)
MSE: 0.00029
MSE: 0.00034
Average Consumption (kWh)
Temperature (C°)

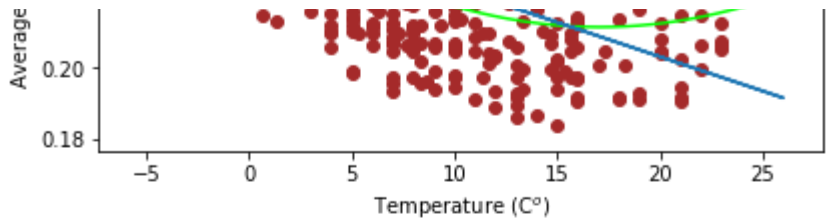

Time: 04:00 (non-TOU)
MSE: 0.00014
MSE: 0.00021
Average Consumption (kWh)
Temperature (C°)

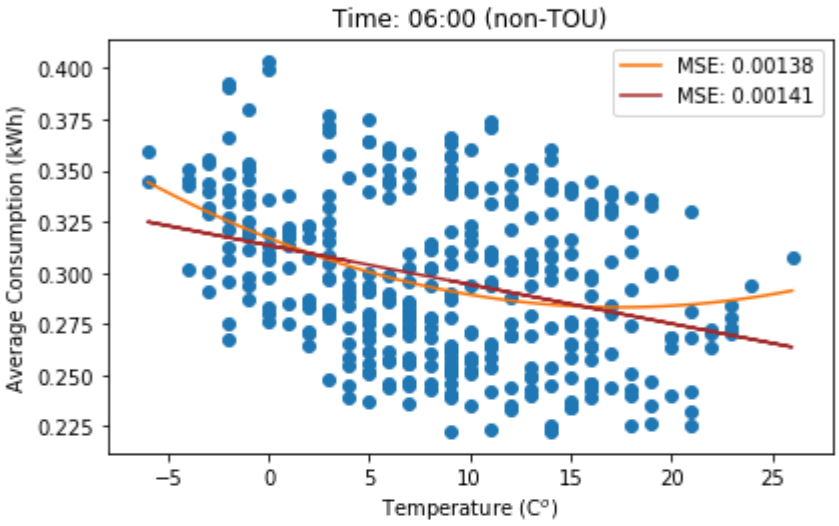

Time: 04:00 (TOU)
MSE: 0.00016
MSE: 0.00021
Average Consumption (kWh)
Temperature (C°)

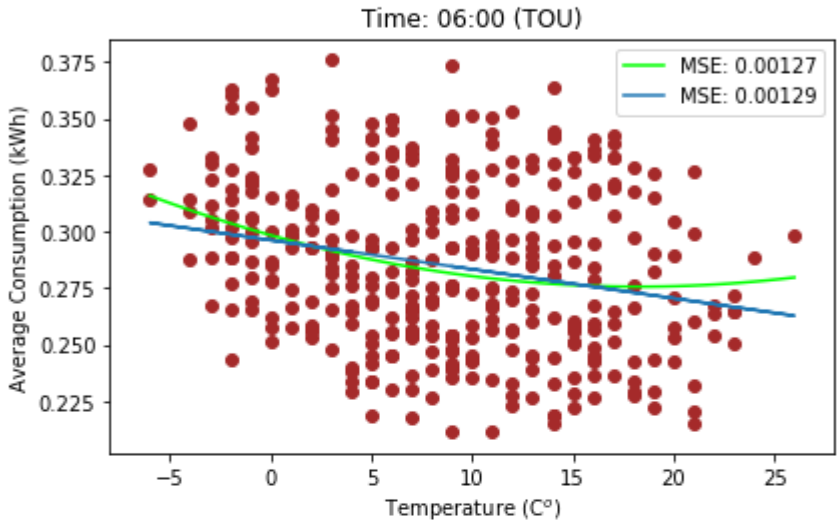

Time: 05:00 (non-TOU)
MSE: 0.00023
MSE: 0.00029
Temperature (C°)

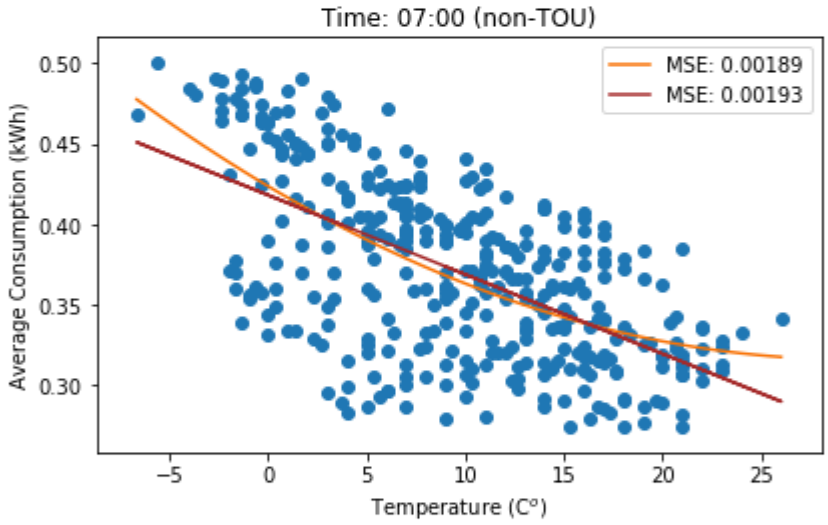

Time: 05:00 (TOU)
MSE: 0.00024
MSE: 0.00027
Temperature (C°)

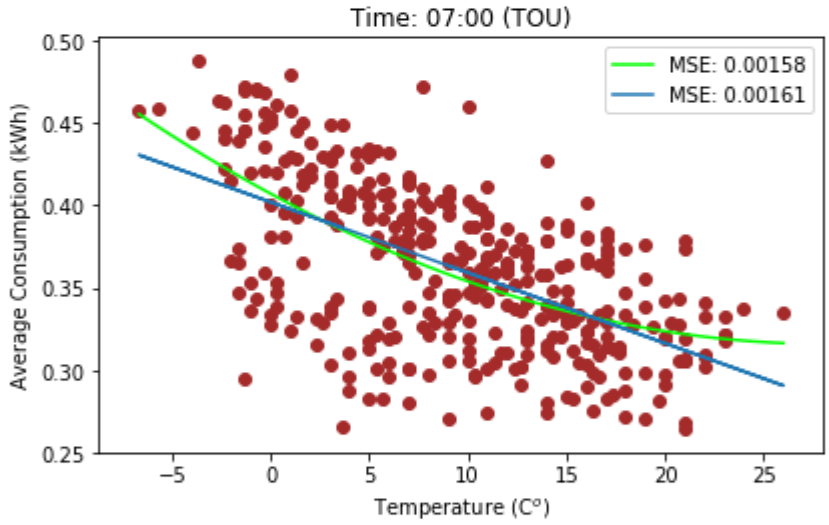

Average
0.22
0.20
−5
0
5
10
15
20
25
Temperature (C°)

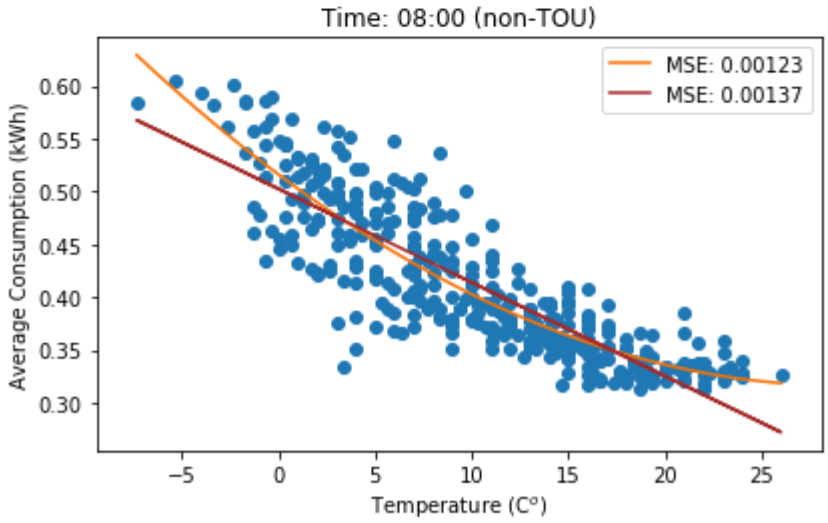

Average
0.20
0.18
−5
0
5
10
15
20
25
Temperature (C°)

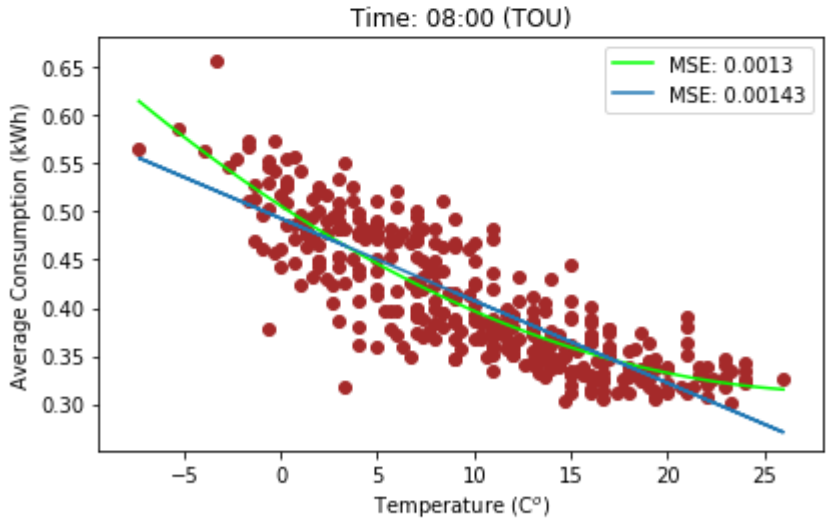

Time: 06:00 (non-TOU)
MSE: 0.00138
MSE: 0.00141
Average Consumption (kWh)
0.400
0.375
0.350
0.325
0.300
0.275
0.250
0.225
−5
0
5
10
15
20
25
Temperature (C°)

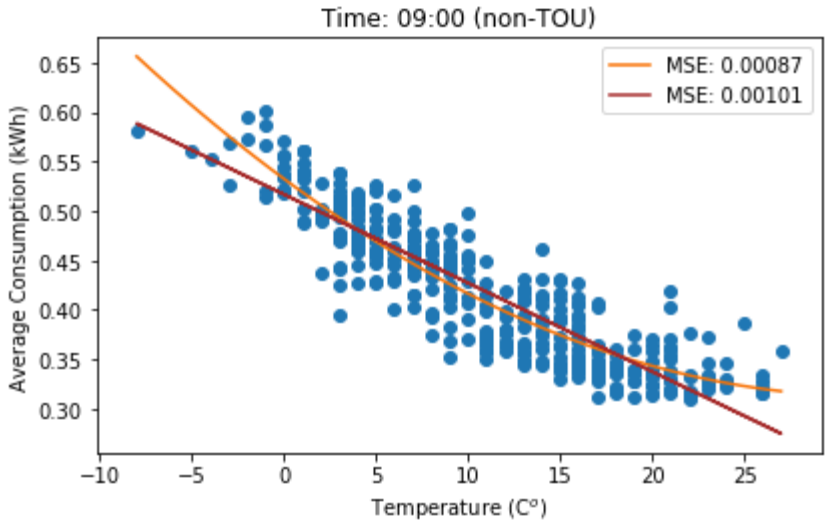

Time: 06:00 (TOU)
MSE: 0.00127
MSE: 0.00129
Average Consumption (kWh)
0.375
0.350
0.325
0.300
0.275
0.250
0.225
−5
0
5
10
15
20
25
Temperature (C°)

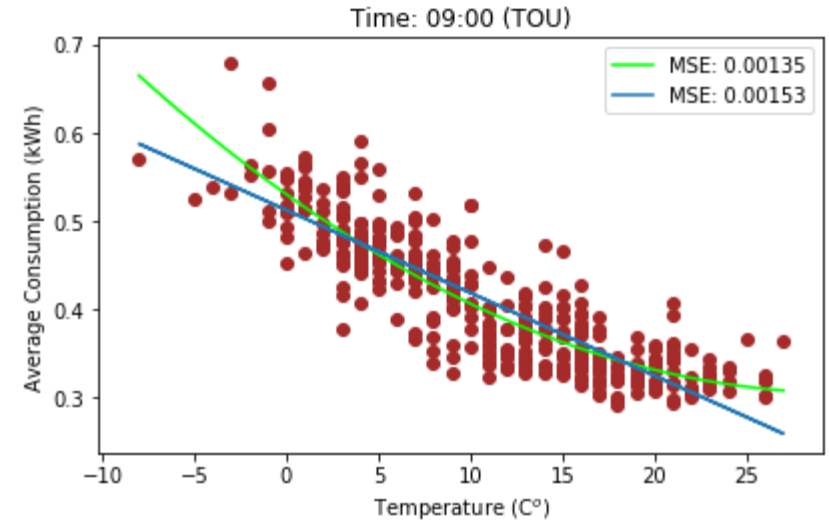

Time: 07:00 (non-TOU)
MSE: 0.00189
MSE: 0.00193
Average Consumption (kWh)
0.50
0.45
0.40
0.35
0.30
−5
0
5
10
15
20
25
Temperature (C°)

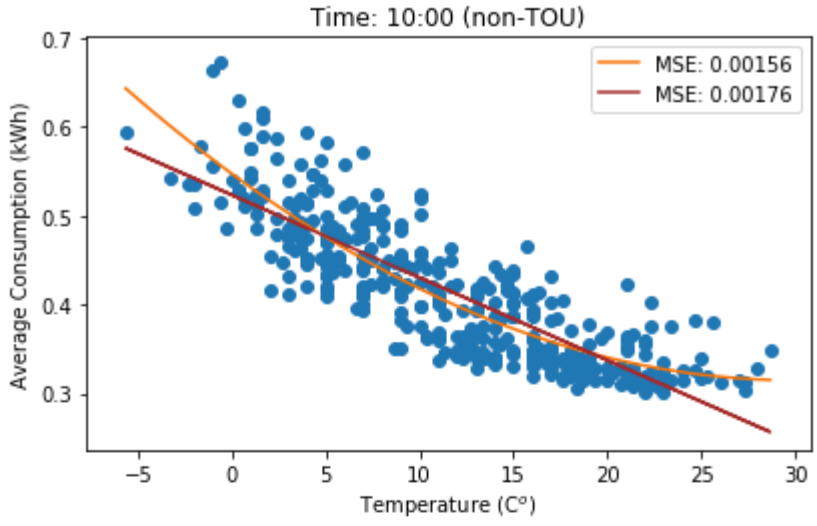

Time: 07:00 (TOU)
MSE: 0.00158
MSE: 0.00161
Average Consumption (kWh)
0.50
0.45
0.40
0.35
0.30
0.25
−5
0
5
10
15
20
25
Temperature (C°)

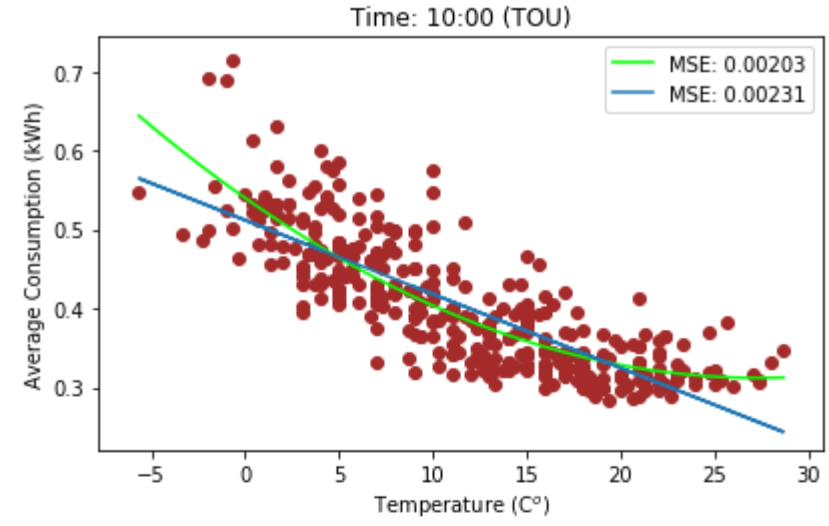

Time: 08:00 (non-TOU)
MSE: 0.00123
MSE: 0.00137
Average Consumption (kWh)
0.60
0.55
0.50
0.45
0.40
0.35
0.30
−5
0
5
10
15
20
25
Temperature (C°)

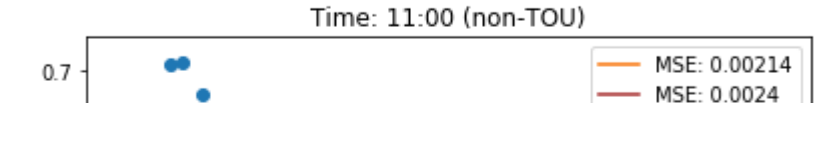

Time: 08:00 (TOU)
MSE: 0.0013
MSE: 0.00143
Average Consumption (kWh)
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
−5
0
5
10
15
20
25
Temperature (C°)

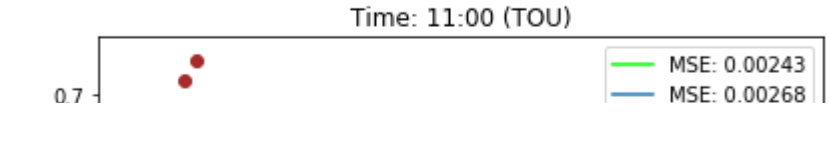

Time: 09:00 (non-TOU)
MSE: 0.00087
MSE: 0.00101
Average Consumption (kWh)
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
−10
−5
0
5
10
15
20
25
Temperature (C°)

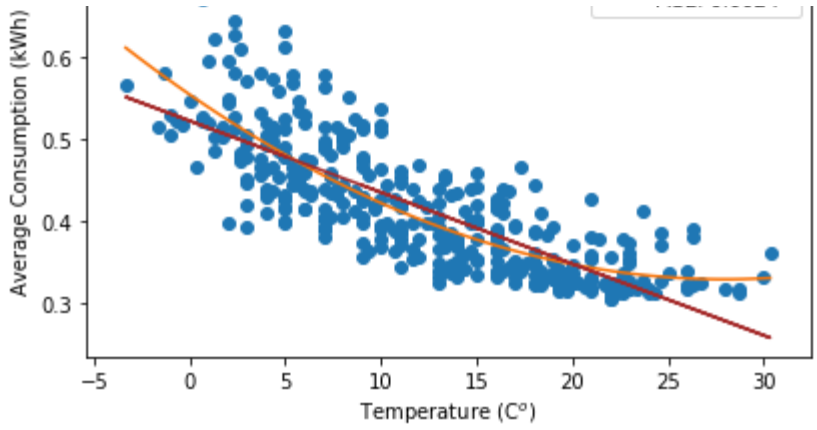

Time: 09:00 (TOU)
MSE: 0.00135
MSE: 0.00153
Average Consumption (kWh)
0.7
0.6
0.5
0.4
0.3
−10
−5
0
5
10
15
20
25
Temperature (C°)

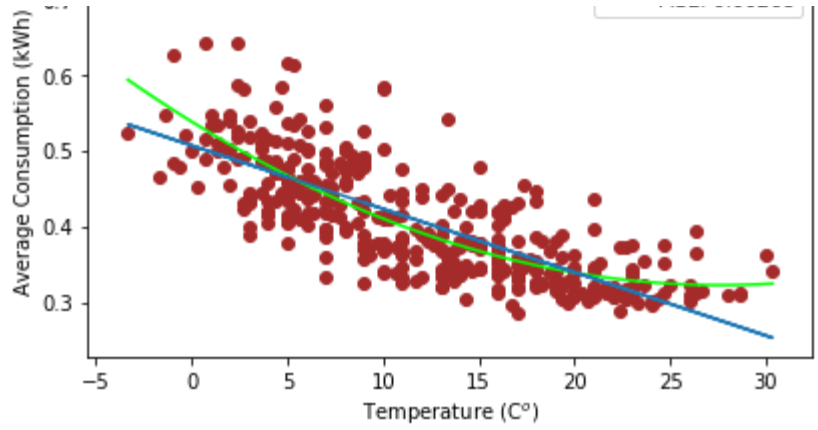

Time: 10:00 (non-TOU)
MSE: 0.00156
MSE: 0.00176
Average Consumption (kWh)
0.7
0.6
0.5
0.4
0.3
−5
0
5
10
15
20
25
30
Temperature (C°)

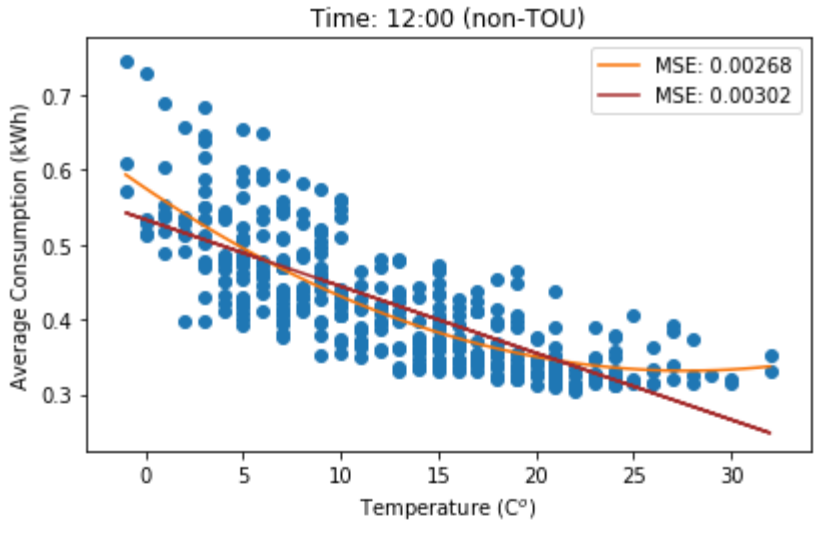

Time: 10:00 (TOU)
MSE: 0.00203
MSE: 0.00231
Average Consumption (kWh)
0.7
0.6
0.5
0.4
0.3
−5
0
5
10
15
20
25
30
Temperature (C°)

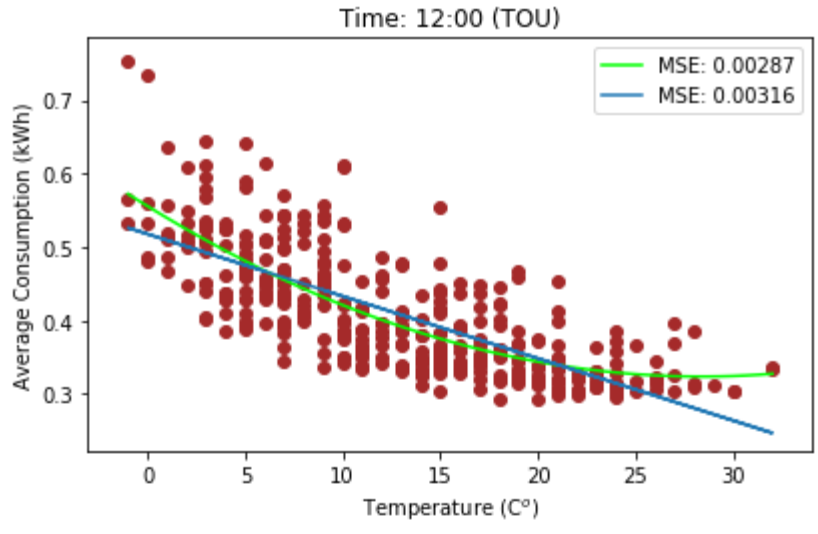

Time: 11:00 (non-TOU)
MSE: 0.00214
MSE: 0.0024
0.7

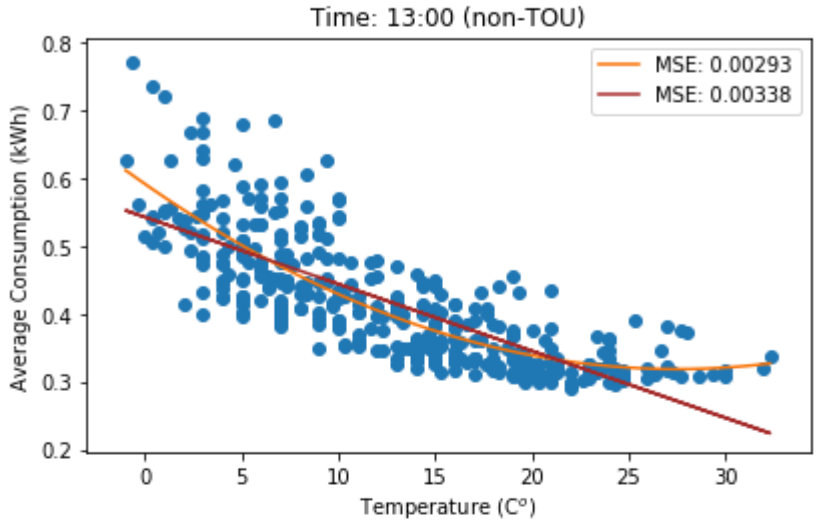

Time: 11:00 (TOU)
MSE: 0.00243
MSE: 0.00268
0.7

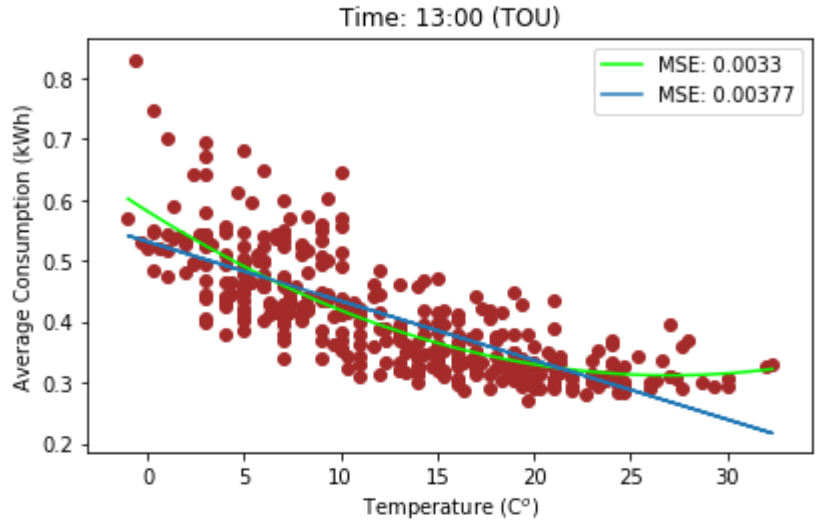

Average Consumption (kWh)
Temperature (C°)

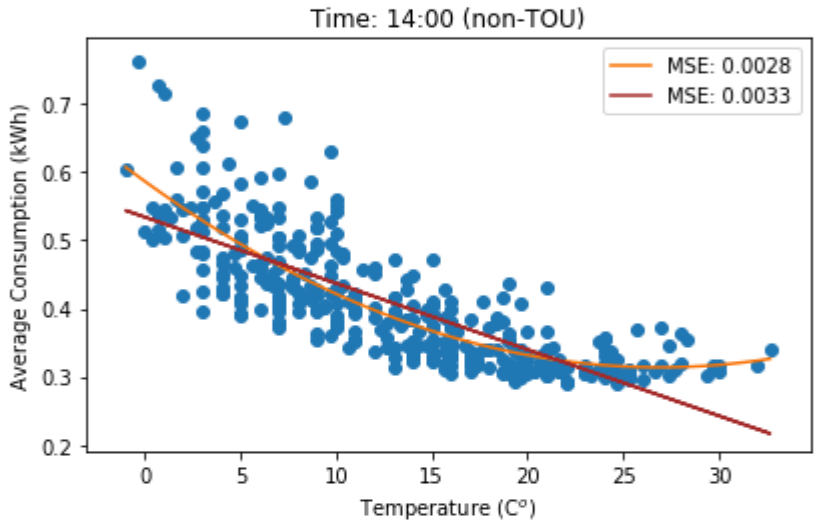

Average Consumption (kWh)
Temperature (C°)

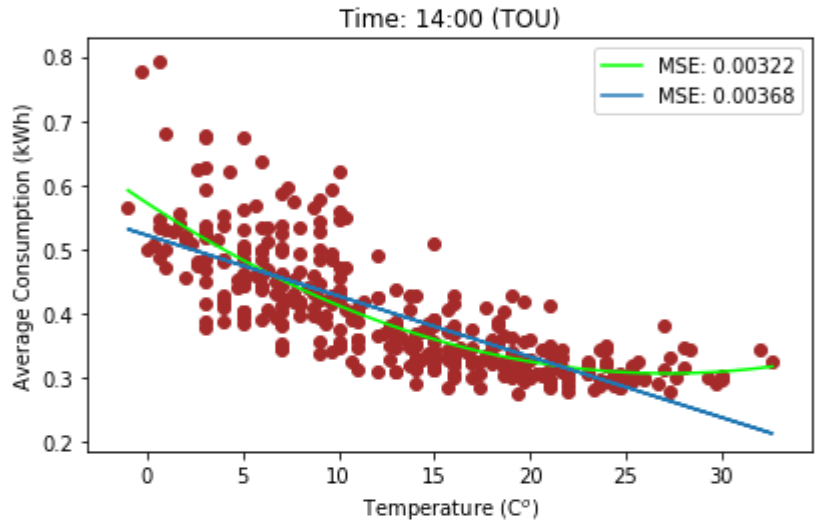

Time: 12:00 (non-TOU)
MSE: 0.00268
MSE: 0.00302
Average Consumption (kWh)
Temperature (C°)

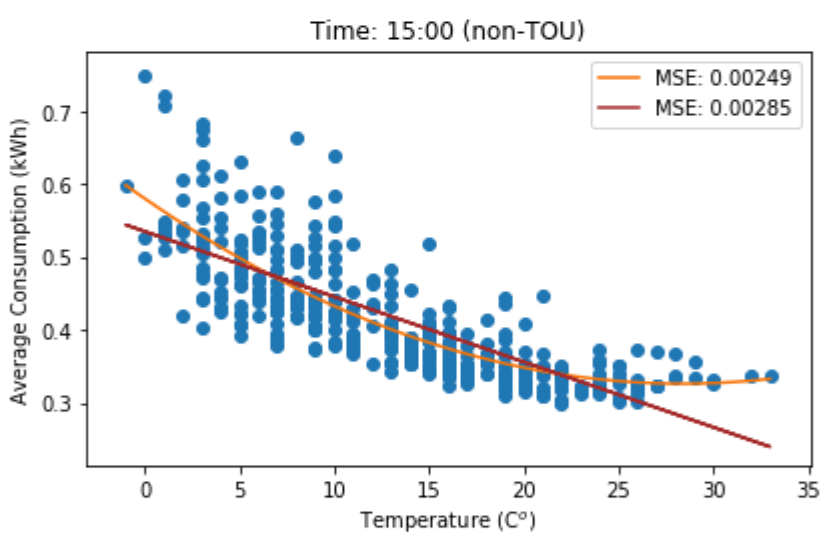

Time: 12:00 (TOU)
MSE: 0.00287
MSE: 0.00316
Average Consumption (kWh)
Temperature (C°)

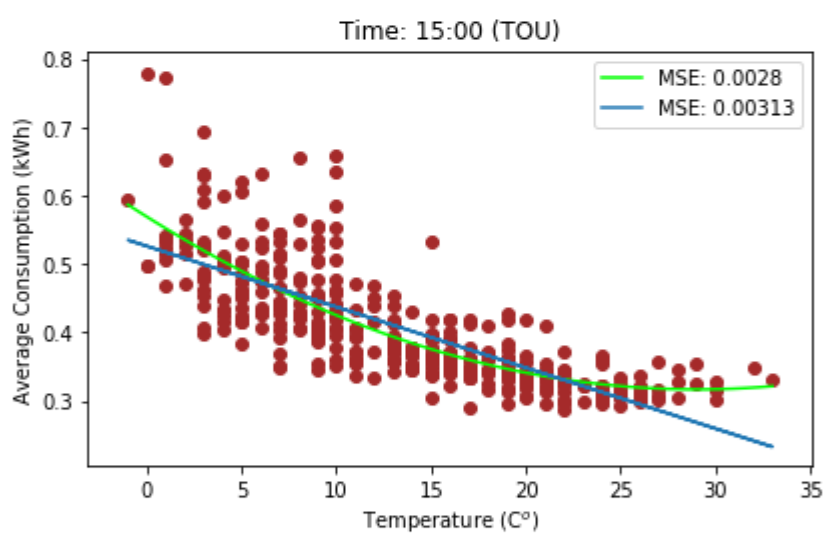

Time: 13:00 (non-TOU)
MSE: 0.00293
MSE: 0.00338
Average Consumption (kWh)
Temperature (C°)

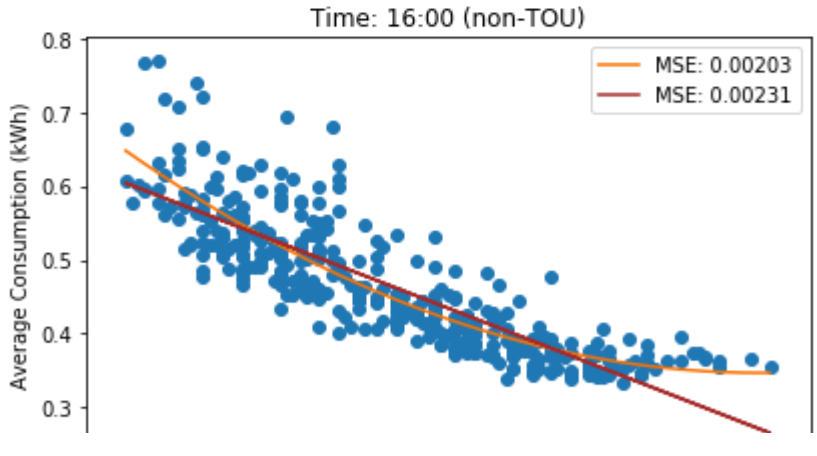

Time: 13:00 (TOU)
MSE: 0.0033
MSE: 0.00377
Average Consumption (kWh)
Temperature (C°)

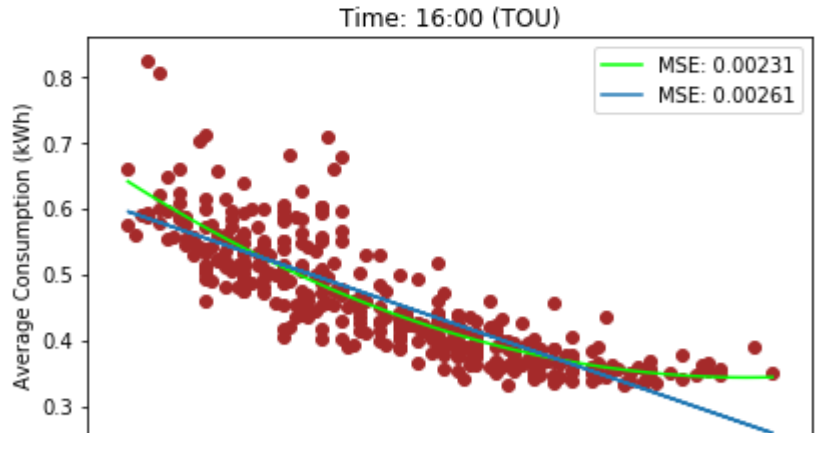

Time: 14:00 (non-TOU)
MSE: 0.0028
MSE: 0.0033
Average Consumption (kWh)
Temperature (C°)

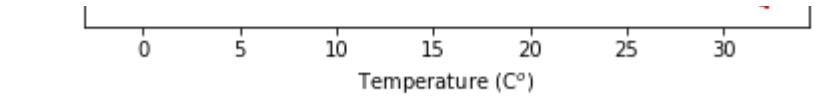

Time: 14:00 (TOU)
MSE: 0.00322
MSE: 0.00368
Average Consumption (kWh)
Temperature (C°)

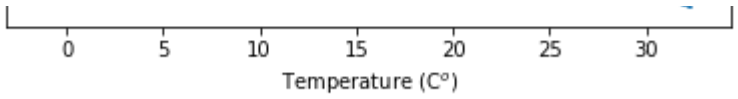

Time: 15:00 (non-TOU)
MSE: 0.00249
MSE: 0.00285
Average Consumption (kWh)
Temperature (C°)

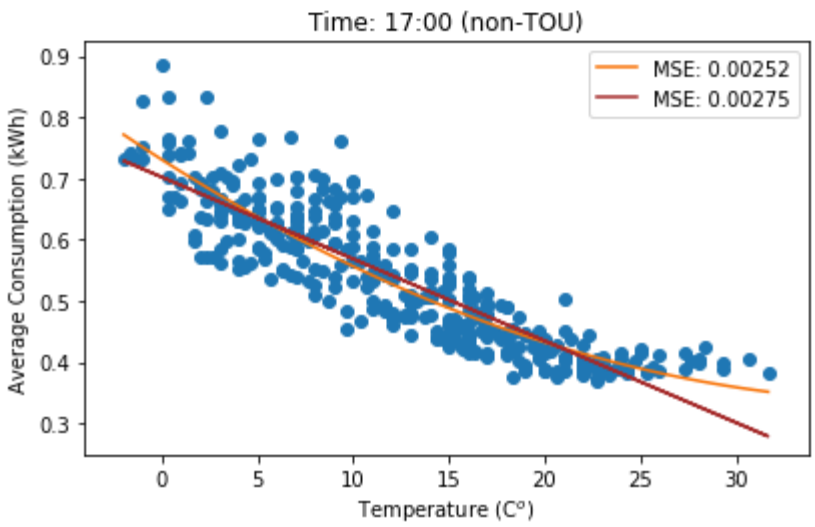

Time: 15:00 (TOU)
MSE: 0.0028
MSE: 0.00313
Average Consumption (kWh)
Temperature (C°)

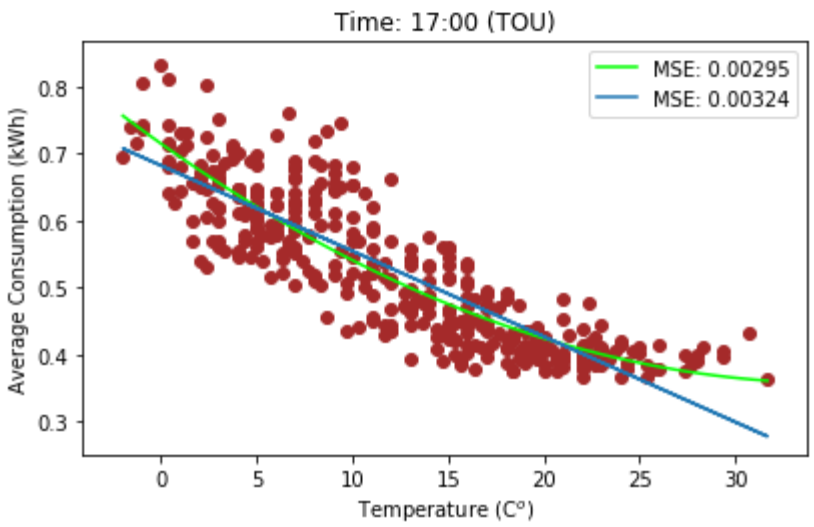

Time: 16:00 (non-TOU)
MSE: 0.00203
MSE: 0.00231
Average Consumption (kWh)

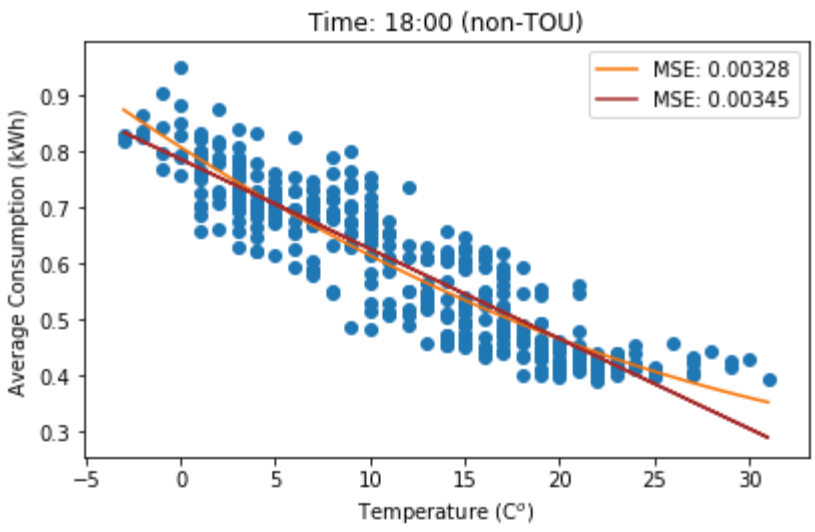

Time: 16:00 (TOU)
MSE: 0.00231
MSE: 0.00261
Average Consumption (kWh)

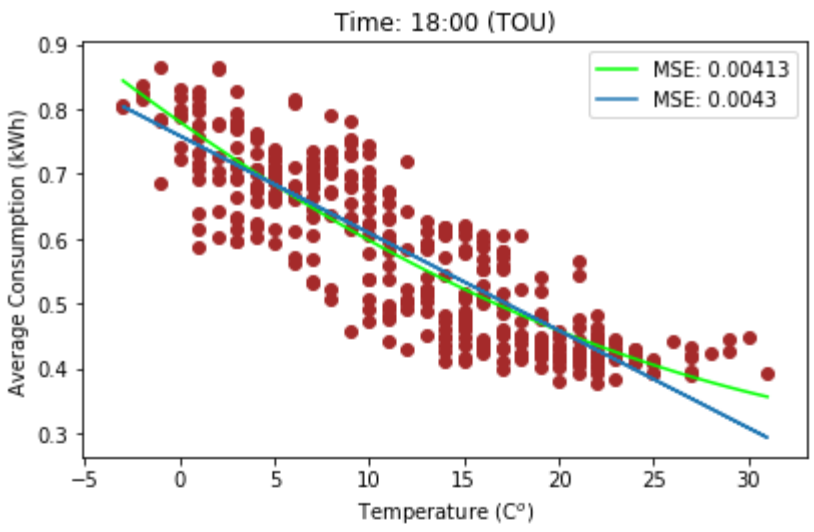

0
5
10
15
20
25
30
Temperature (C°)

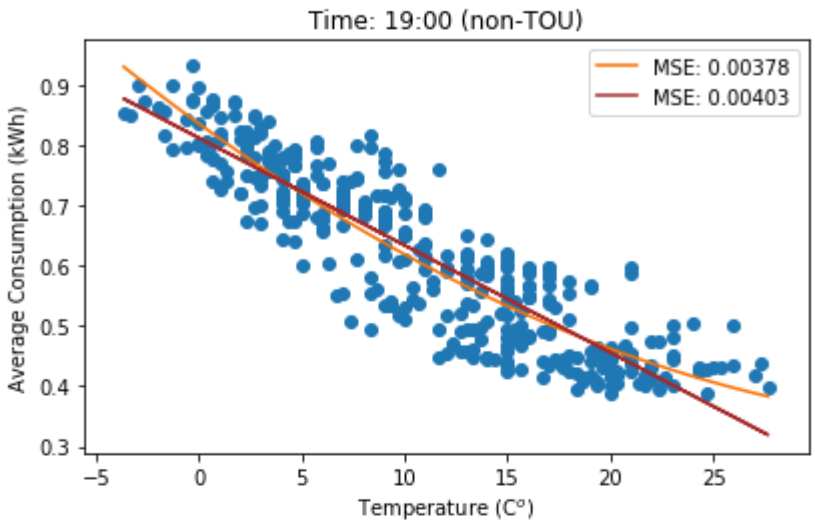

0
5
10
15
20
25
30
Temperature (C°)

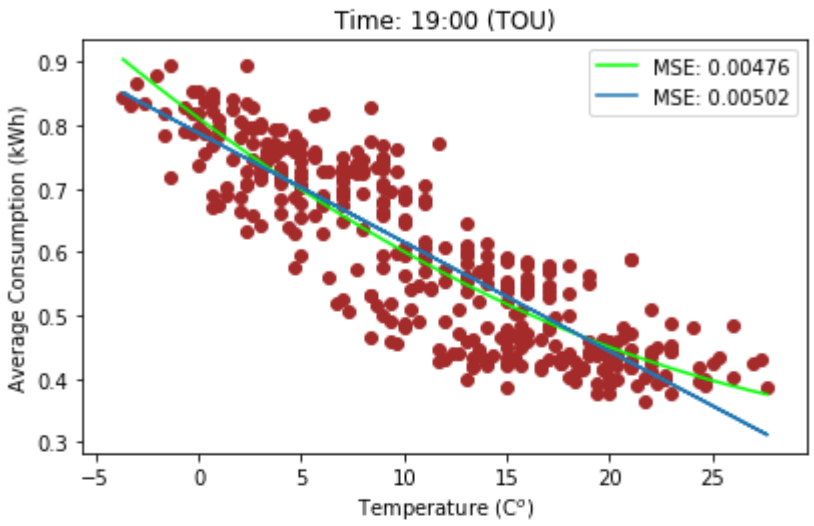

Time: 17:00 (non-TOU)
MSE: 0.00252
MSE: 0.00275
Average Consumption (kWh)
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
5
10
15
20
25
30
Temperature (C°)

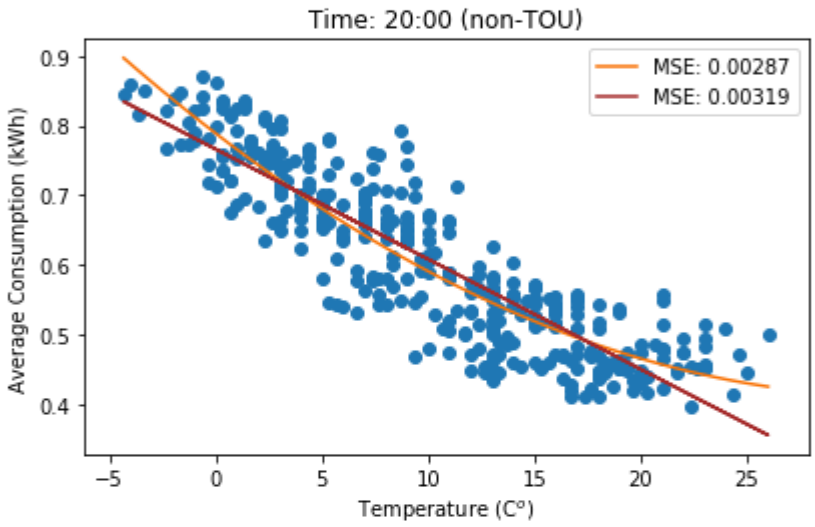

Time: 17:00 (TOU)
MSE: 0.00295
MSE: 0.00324
Average Consumption (kWh)
0.8
0.7
0.6
0.5
0.4
0.3
0
5
10
15
20
25
30
Temperature (C°)

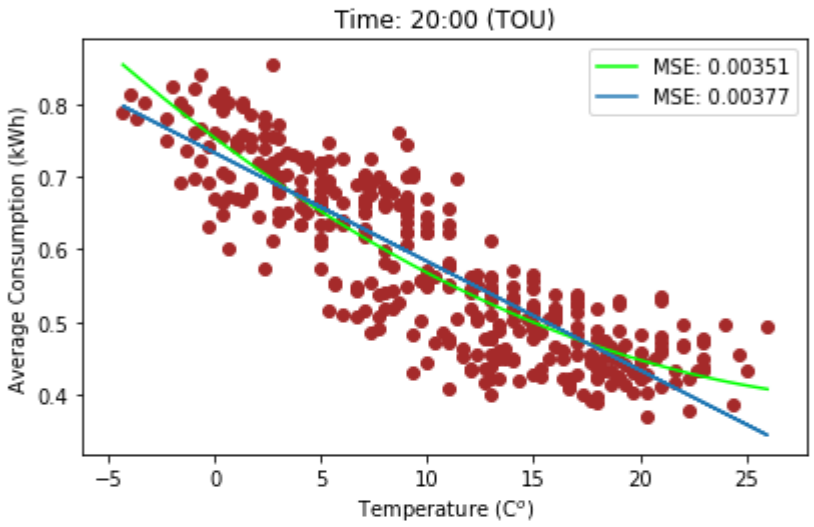

Time: 18:00 (non-TOU)
MSE: 0.00328
MSE: 0.00345
Average Consumption (kWh)
0.9
0.8
0.7
0.6
0.5
0.4
0.3
−5
0
5
10
15
20
25
30
Temperature (C°)

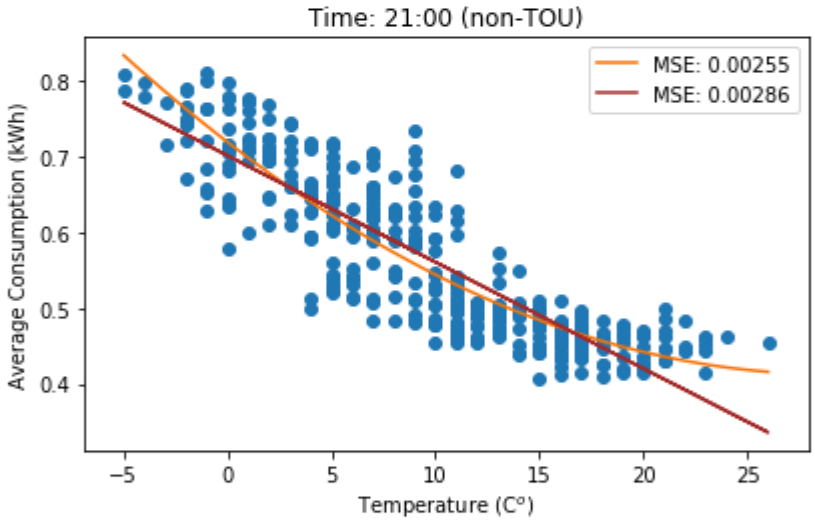

Time: 18:00 (TOU)
MSE: 0.00413
MSE: 0.0043
Average Consumption (kWh)
0.9
0.8
0.7
0.6
0.5
0.4
0.3
−5
0
5
10
15
20
25
30
Temperature (C°)

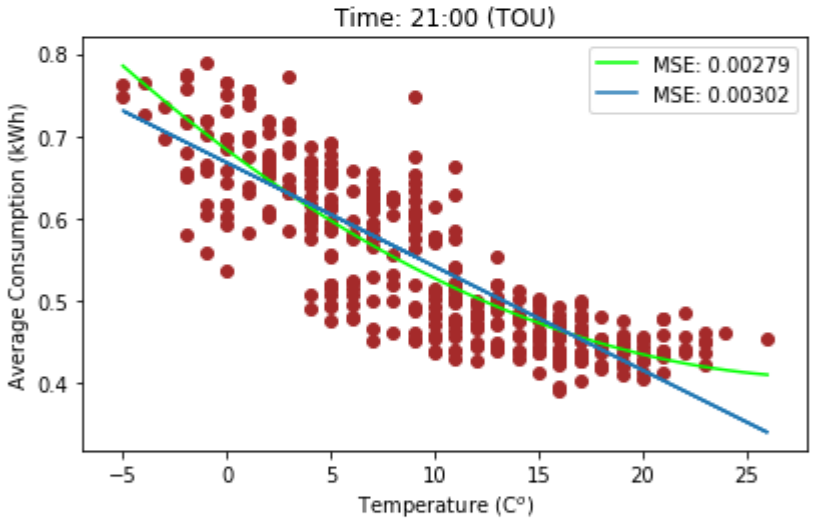

Time: 19:00 (non-TOU)
MSE: 0.00378
MSE: 0.00403
Average Consumption (kWh)
0.9
0.8
0.7
0.6
0.5
0.4
0.3
−5
0
5
10
15
20
25
Temperature (C°)

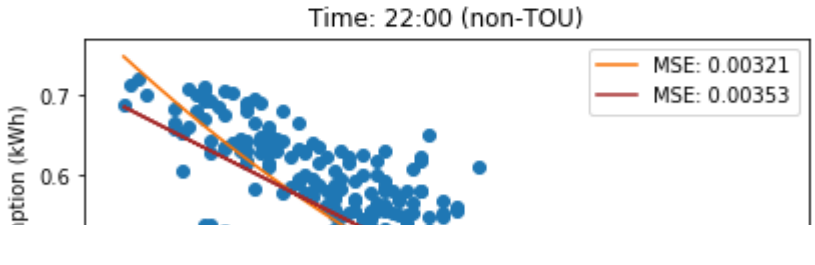

Time: 19:00 (TOU)
MSE: 0.00476
MSE: 0.00502
Average Consumption (kWh)
0.9
0.8
0.7
0.6
0.5
0.4
0.3
−5
0
5
10
15
20
25
Temperature (C°)

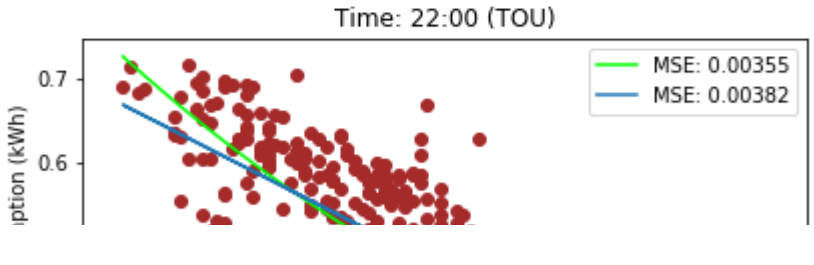

Time: 20:00 (non-TOU)
MSE: 0.00287
MSE: 0.00319
Average Consumption (kWh)
0.9
0.8
0.7
0.6
0.5
0.4
−5
0
5
10
15
20
25
Temperature (C°)

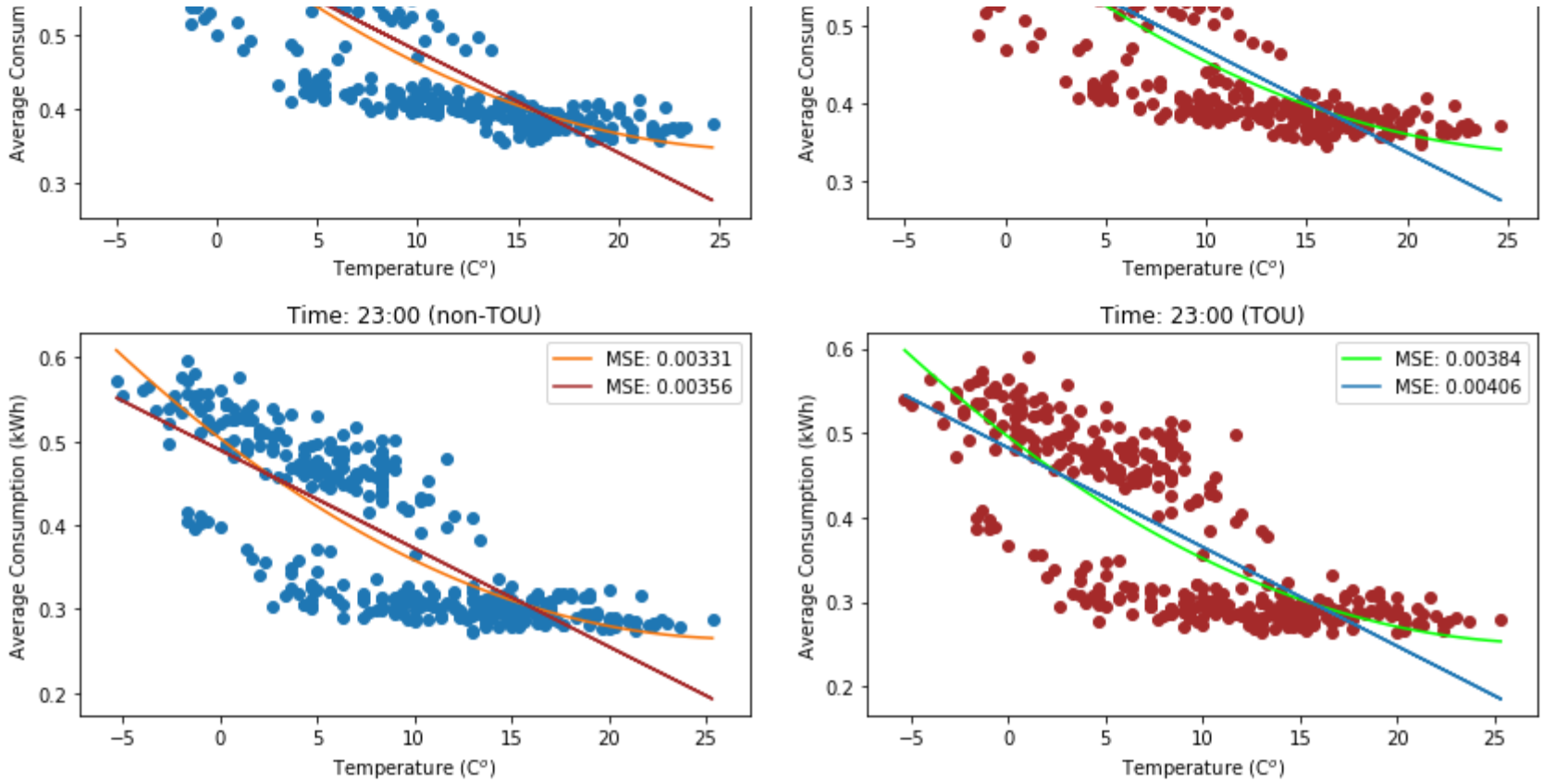

Time: 20:00 (TOU)
MSE: 0.00351
MSE: 0.00377
Average Consumption (kWh)
0.8
0.7
0.6
0.5
0.4
−5
0
5
10
15
20
25
Temperature (C°)


Time: 21:00 (non-TOU)
MSE: 0.00255
MSE: 0.00286
Average Consumption (kWh)
0.8
0.7
0.6
0.5
0.4
−5
0
5
10
15
20
25
Temperature (C°)


Time: 21:00 (TOU)
MSE: 0.00279
MSE: 0.00302
Average Consumption (kWh)
0.8
0.7
0.6
0.5
0.4
−5
0
5
10
15
20
25
Temperature (C°)


Time: 22:00 (non-TOU)
MSE: 0.00321
MSE: 0.00353
0.7
0.6


Time: 22:00 (TOU)
MSE: 0.00355
MSE: 0.00382
0.7
0.6



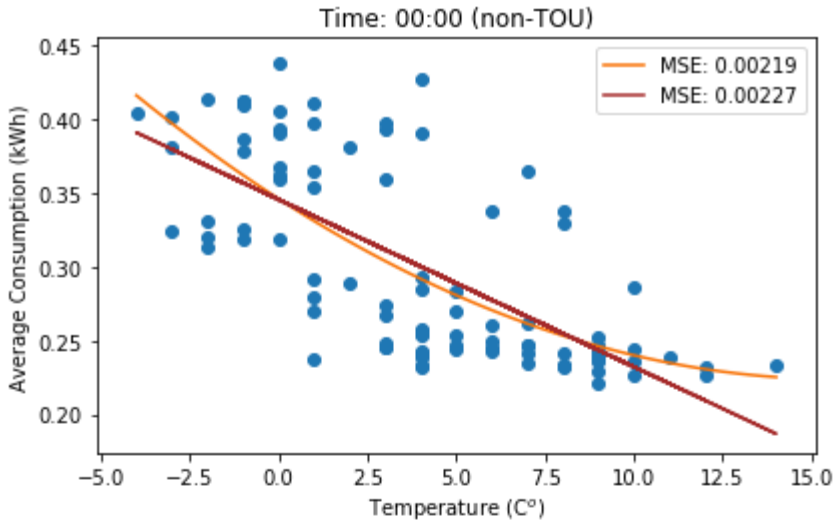
Monthly characteristics

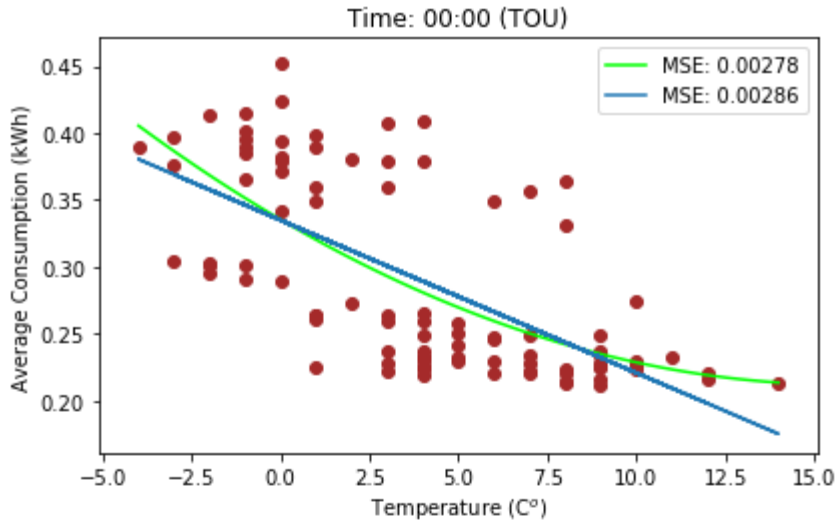
Notice that these data points have certain gaps, especially for 22:00 to 04:00. So we wonder if that's due to seasonal effects. London seasons are listed below: Spring (March - May), Summer (June - August), Autumn (September - November), and Winter (December - February)

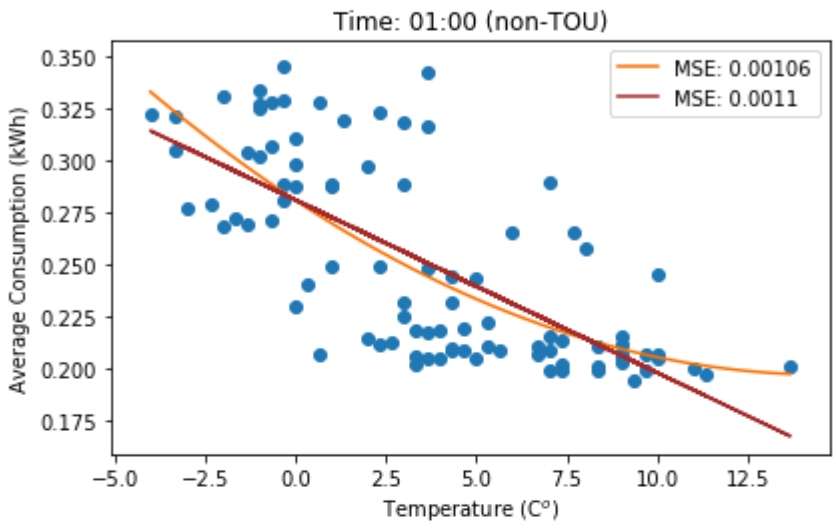
In [7]:
```
# Read the tariff data
df_tariff = pd.read_csv("C:\\Users\\Rockwell\\Desktop\\Paper4\\data_collection\\data_tables\\tariff_d.csv")

# Shift the Price and Events Column up by 1 row (30min)
df_tariff.Price = df_tariff.Price.shift(-1)
df_tariff.Event_tags = df_tariff.Event_tags.shift(-1)

# Add the first row as starting from '2013-01-01 00:00:00'
new_row = pd.DataFrame({'GMT' : ['2013-01-01 00:00:00'],
                        'Price' : [np.float64(0.1176)],
                        'Event_tags' : float('nan')})
df_tariff_edited = pd.concat([new_row, df_tariff], ignore_index = True)[:-11]

# make the frequency to one hour
df_tariff_1h = df_tariff_edited[::2].reset_index(drop = True)
df_tariff_1h.to_csv("C:\\Users\\Rockwell\\Desktop\\Paper4\\data_collection\\data_tables\\df_tariff_1h.csv", i
ndex = False)
```

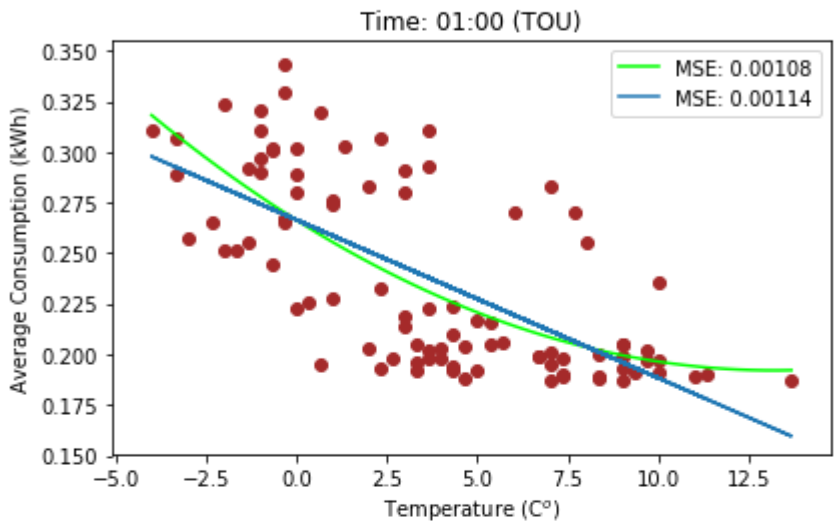
In [14]:
```
# Create Spring, Summer, Autumn and Winter dataframe for load, price, weather
spring = [3, 4, 5]
summer = [6, 7, 8]
autumn = [9, 10, 11]
winter = [12, 1, 2]
df_month_help = pd.DataFrame(pd.to_datetime(df_tariff_1h.GMT).dt.month)
df_wea1h_spring = df_wea1h[df_month_help['GMT'].isin(spring)]
df_wea1h_summer = df_wea1h[df_month_help['GMT'].isin(summer)]
df_wea1h_autumn = df_wea1h[df_month_help['GMT'].isin(autumn)]
df_wea1h_winter = df_wea1h[df_month_help['GMT'].isin(winter)]

df_tou1h_spring = df_tou1h[df_month_help['GMT'].isin(spring)]
df_tou1h_summer = df_tou1h[df_month_help['GMT'].isin(summer)]
df_tou1h_autumn = df_tou1h[df_month_help['GMT'].isin(autumn)]
df_tou1h_winter = df_tou1h[df_month_help['GMT'].isin(winter)]

df_Ntou1h_spring = df_Ntou1h[df_month_help['GMT'].isin(spring)]
df_Ntou1h_summer = df_Ntou1h[df_month_help['GMT'].isin(summer)]
df_Ntou1h_autumn = df_Ntou1h[df_month_help['GMT'].isin(autumn)]
df_Ntou1h_winter = df_Ntou1h[df_month_help['GMT'].isin(winter)]
```

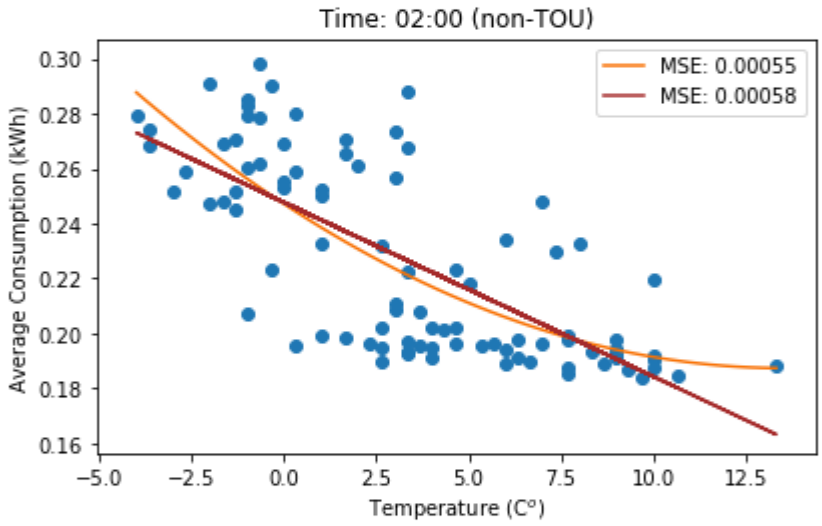
In [19]:
```
# Spring scatterplot of average consumption (over all consumers at same time period) vs. temp
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
# non-TOU vs TOU in terms of showing the elements in Average column such that they are hourly energy consumpt
ion v.s. temperature over 365 days
fig_all = plt.figure(figsize = (12,90))
ax_Ntou = [] # store subplot objects
ax_tou = [] # store subplot objects
lm = LinearRegression()
for i in range(24):
    ax_Ntou.append(fig_all.add_subplot(24, 2, 2 * i + 1))
    if i <= 9:
        x = df_wea1h_spring.TempC[df_wea1h_spring.GMT.str.contains('0' + str(i) + ':00:00')].values
        y = df_Ntou1h_spring.Average[df_Ntou1h_spring.GMT.str.contains('0' + str(i) + ':00:00')].values
        xp = np.linspace(min(x), max(x), 50) # polynomial regression
        z = np.polyfit(x, y, 2)
        p = np.poly1d(z)
        ax_Ntou[-1].plot(xp, p(xp), color = 'xkcd:orange', label = 'MSE: ' + str(round(mean_squared_error(y,
p(x)), 5)))
        x = x.reshape(-1, 1)
        y = y.reshape(-1, 1)
        lm.fit(x, y)
        ax_Ntou[-1].plot(x, lm.predict(x), color = 'brown', label = 'MSE: ' + str(round(mean_squared_error(y,
lm.predict(x)), 5)))
        ax_Ntou[-1].scatter(x, y)
        ax_Ntou[-1].set_title('Time: 0' + str(i) + ':00 (non-TOU)')
        ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_Ntou[-1].set_ylabel('Average Consumption (kWh)')
        ax_Ntou[-1].legend()
    else:
        x = df_wea1h_spring.TempC[df_wea1h_spring.GMT.str.contains(str(i) + ':00:00')].values
        y = df_Ntou1h_spring.Average[df_Ntou1h_spring.GMT.str.contains(str(i) + ':00:00')].values
        xp = np.linspace(min(x), max(x), 50) # polynomial regression
        z = np.polyfit(x, y, 2)
        p = np.poly1d(z)
        ax_Ntou[-1].plot(xp, p(xp), color = 'xkcd:orange', label = 'MSE: ' + str(round(mean_squared_error(y,
p(x)), 5)))
        x = x.reshape(-1, 1)
        y = y.reshape(-1, 1)
        lm.fit(x, y)
        ax_Ntou[-1].plot(x, lm.predict(x), color = 'brown', label = 'MSE: ' + str(round(mean_squared_error(y,
lm.predict(x)), 5)))
        ax_Ntou[-1].scatter(x, y)
        ax_Ntou[-1].set_title('Time: ' + str(i) + ':00 (non-TOU)')
        ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_Ntou[-1].set_ylabel('Average Consumption (kWh)')
        ax_Ntou[-1].legend()
for i in range(24):
    ax_tou.append(fig_all.add_subplot(24, 2, 2 * i + 2))
    if i <= 9:
        x = df_wea1h_spring.TempC[df_wea1h_spring.GMT.str.contains('0' + str(i) + ':00:00')].values
        y = df_tou1h_spring.Average[df_tou1h_spring.GMT.str.contains('0' + str(i) + ':00:00')].values
        xp = np.linspace(min(x), max(x), 50) # polynomial regression
        z = np.polyfit(x, y, 2)
        p = np.poly1d(z)
        ax_tou[-1].plot(xp, p(xp), color = 'lime', label = 'MSE: ' + str(round(mean_squared_error(y, p(x)), 5
)))
        x = x.reshape(-1, 1)
        y = y.reshape(-1, 1)
        lm.fit(x, y)
        ax_tou[-1].plot(x, lm.predict(x), label = 'MSE: ' + str(round(mean_squared_error(y, lm.predict(x)), 5
)))
        ax_tou[-1].scatter(x, y, color = 'brown')
        ax_tou[-1].set_title('Time: 0' + str(i) + ':00 (TOU)')
        ax_tou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_tou[-1].set_ylabel('Average Consumption (kWh)')
        ax_tou[-1].legend()
    else:
        x = df_wea1h_spring.TempC[df_wea1h_spring.GMT.str.contains(str(i) + ':00:00')].values
        y = df_tou1h_spring.Average[df_tou1h_spring.GMT.str.contains(str(i) + ':00:00')].values
        xp = np.linspace(min(x), max(x), 50) # polynomial regression
        z = np.polyfit(x, y, 2)
        p = np.poly1d(z)
        ax_tou[-1].plot(xp, p(xp), color = 'lime', label = 'MSE: ' + str(round(mean_squared_error(y, p(x)), 5
)))
        x = x.reshape(-1, 1)
        y = y.reshape(-1, 1)
        lm.fit(x, y)
        ax_tou[-1].plot(x, lm.predict(x), label = 'MSE: ' + str(round(mean_squared_error(y, lm.predict(x)), 5
)))
        ax_tou[-1].scatter(x, y, color = 'brown')
        ax_tou[-1].set_title('Time: ' + str(i) + ':00 (TOU)')
        ax_tou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_tou[-1].set_ylabel('Average Consumption (kWh)')
        ax_tou[-1].legend()
plt.tight_layout()
```

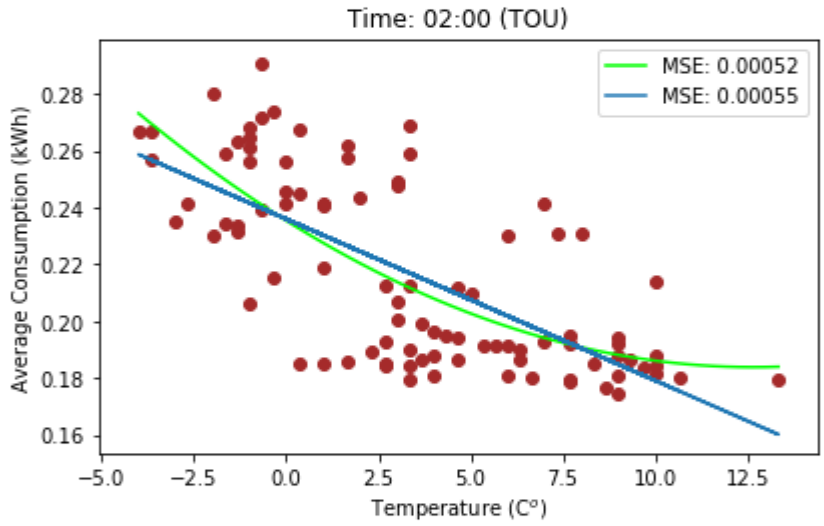

Time: 00:00 (non-TOU)
MSE: 0.00219
MSE: 0.00227
Average Consumption (kWh)
Temperature (C°)

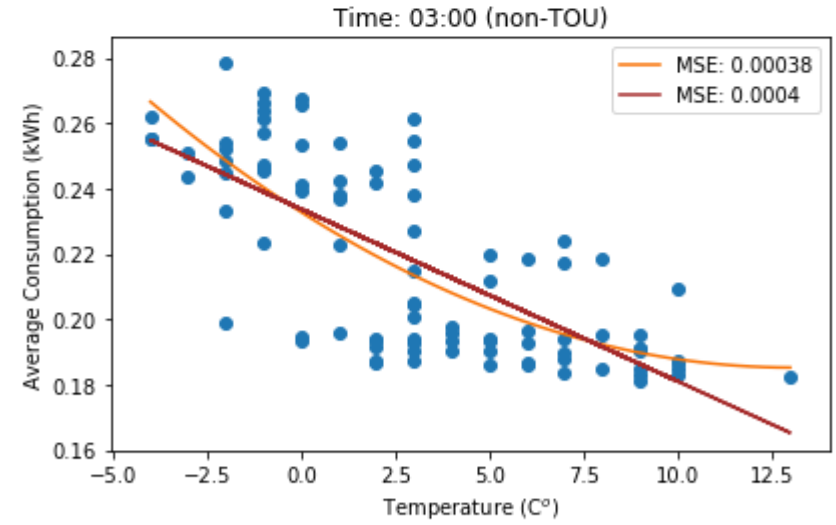

Time: 00:00 (TOU)
MSE: 0.00278
MSE: 0.00286
Average Consumption (kWh)
Temperature (C°)

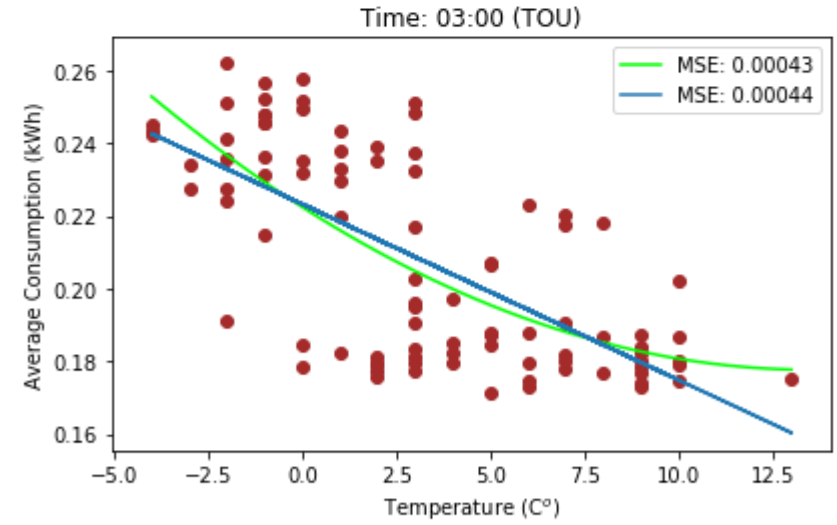

Time: 01:00 (non-TOU)
MSE: 0.00106
MSE: 0.0011
Average Consumption (kWh)
Temperature (C°)

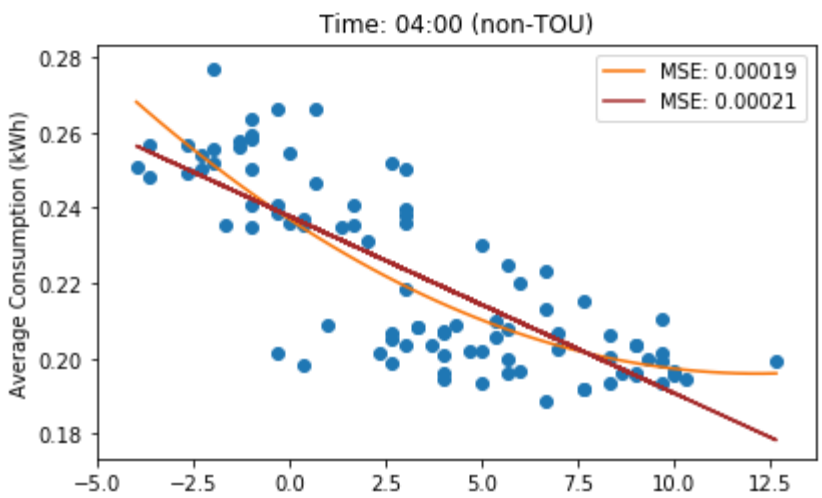

Time: 01:00 (TOU)
MSE: 0.00108
MSE: 0.00114
Average Consumption (kWh)
Temperature (C°)

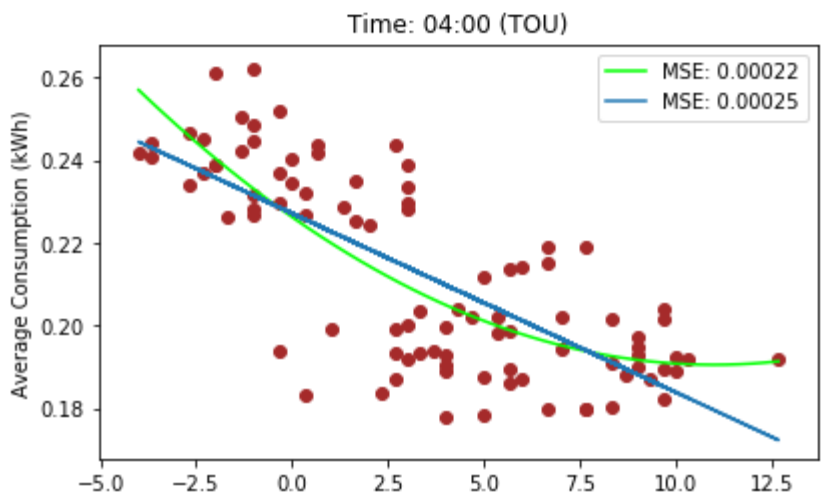

Time: 02:00 (non-TOU)
MSE: 0.00055
MSE: 0.00058
Average Consumption (kWh)
Temperature (C°)


Time: 02:00 (TOU)
MSE: 0.00052
MSE: 0.00055
Average Consumption (kWh)
Temperature (C°)


Time: 03:00 (non-TOU)
MSE: 0.00038
MSE: 0.0004
Average Consumption (kWh)
Temperature (C°)

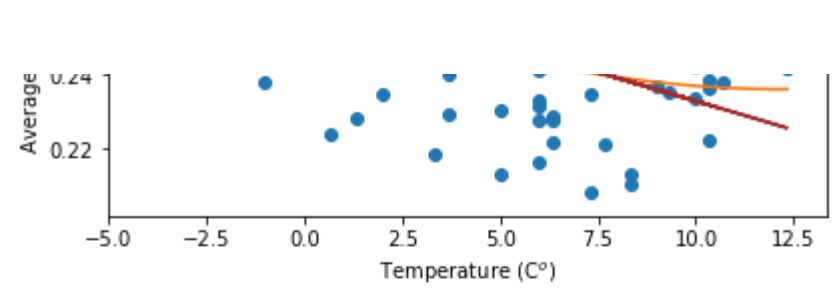

Time: 03:00 (TOU)
MSE: 0.00043
MSE: 0.00044
Average Consumption (kWh)
Temperature (C°)

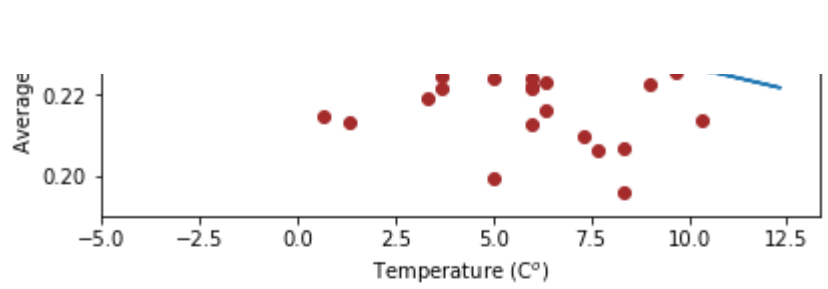

Time: 04:00 (non-TOU)
MSE: 0.00019
MSE: 0.00021
Average Consumption (kWh)
Temperature (C°)

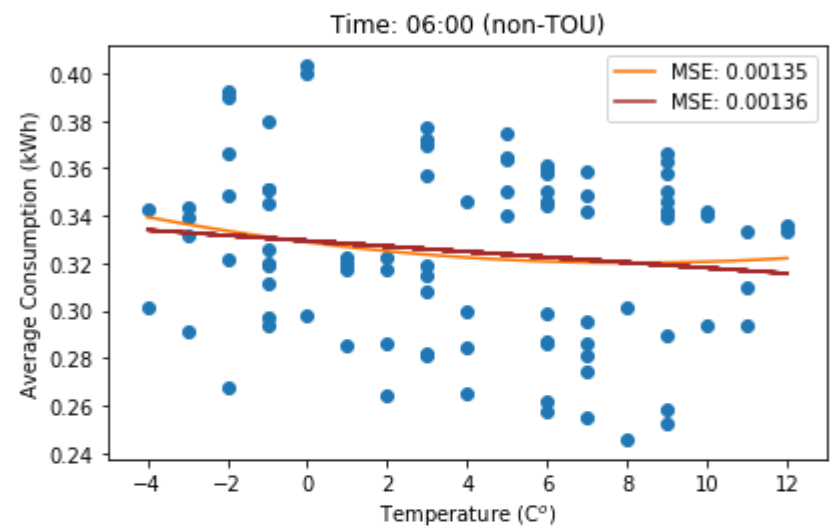

Time: 04:00 (TOU)
MSE: 0.00022
MSE: 0.00025
Average Consumption (kWh)
Temperature (C°)

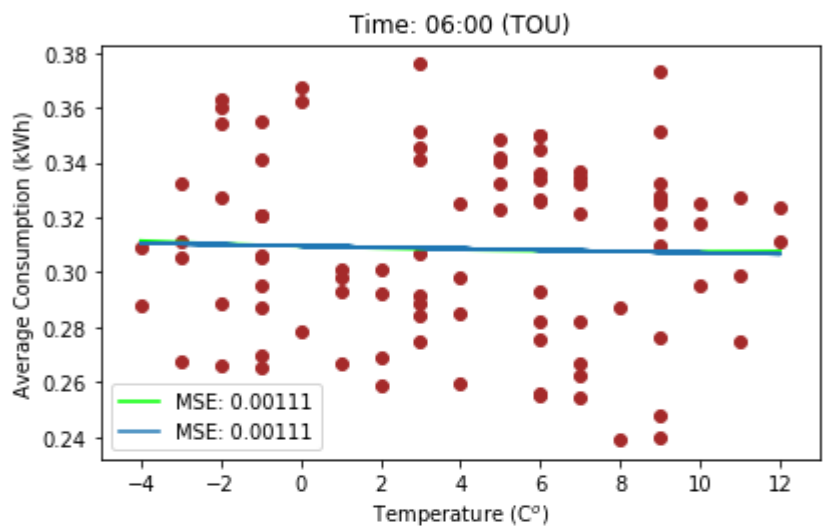
Time: 05:00 (non-TOU)
MSE: 0.0003
MSE: 0.00031
Temperature (C°)

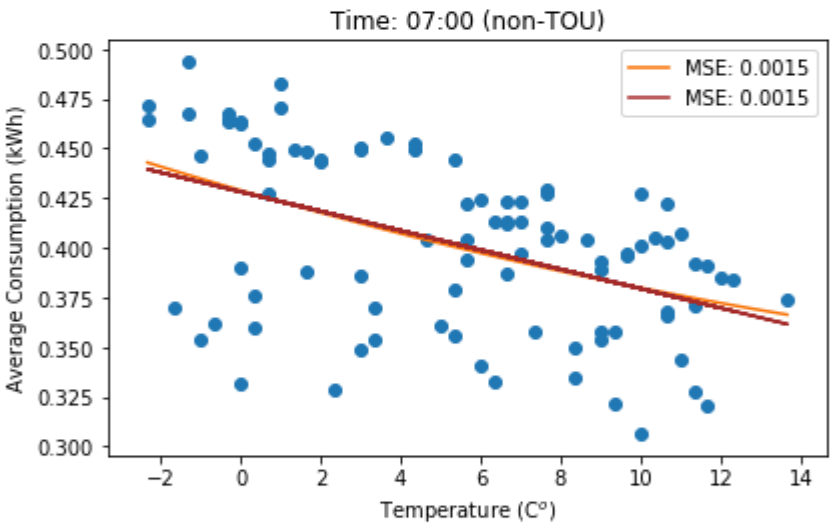
Time: 05:00 (TOU)
MSE: 0.00027
MSE: 0.00027
Temperature (C°)

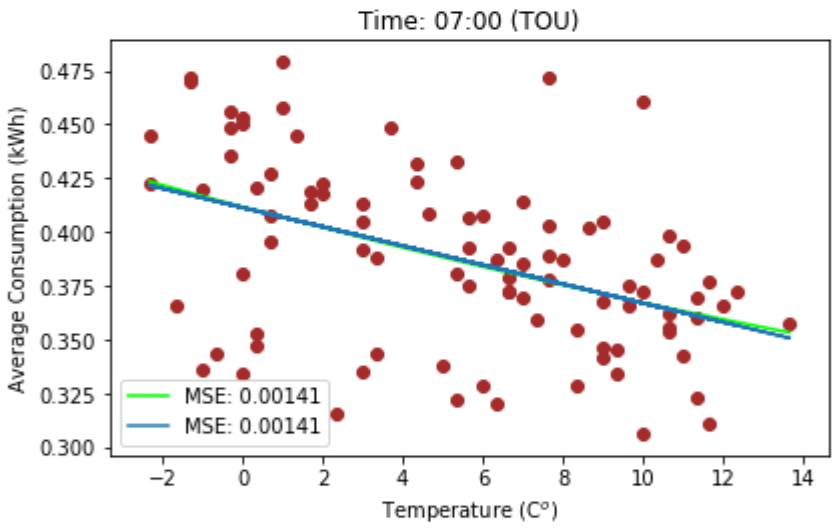

Average
0.24
0.22
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
12.5
Temperature (C°)

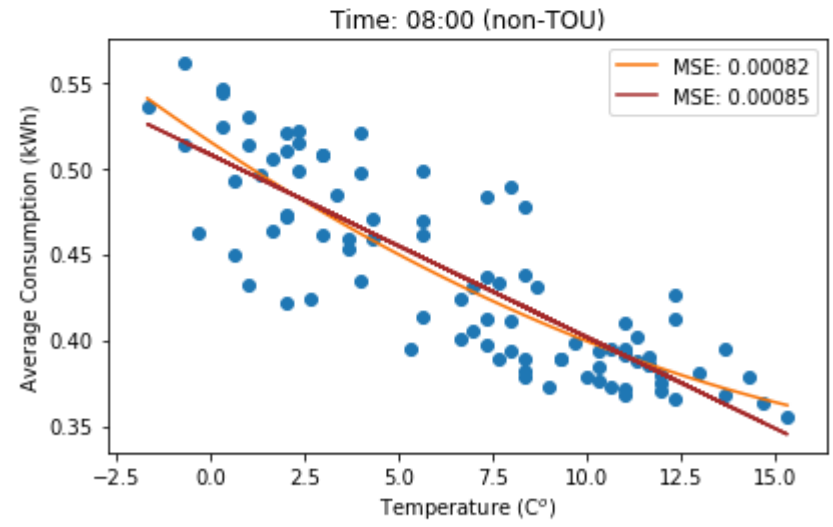

Average
0.22
0.20
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
12.5
Temperature (C°)

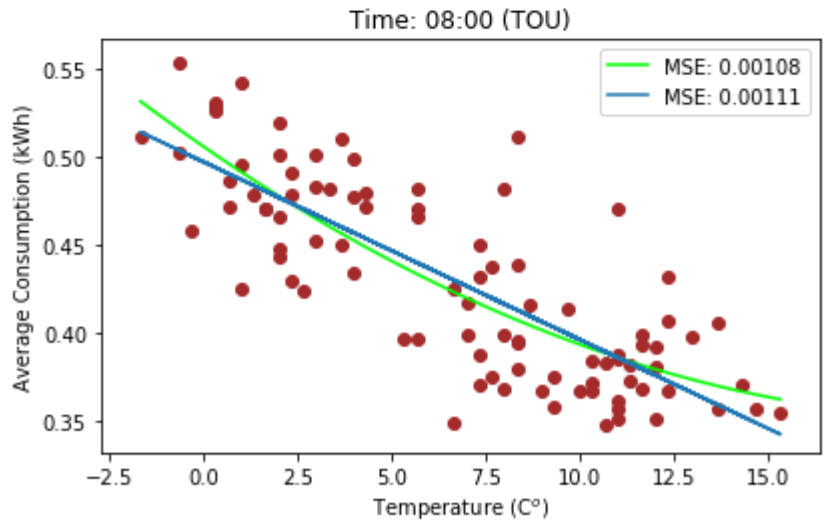

Time: 06:00 (non-TOU)
MSE: 0.00135
MSE: 0.00136
Average Consumption (kWh)
0.40
0.38
0.36
0.34
0.32
0.30
0.28
0.26
0.24
−4
−2
0
2
4
6
8
10
12
Temperature (C°)

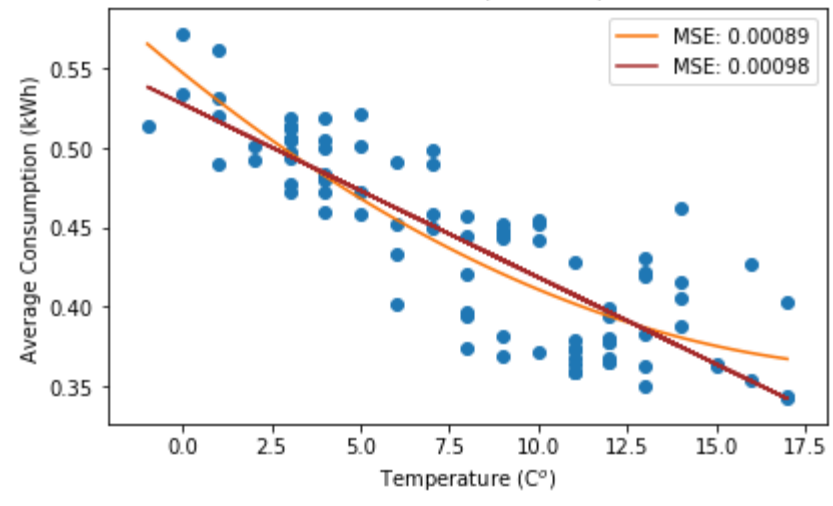

Time: 06:00 (TOU)
MSE: 0.00111
MSE: 0.00111
Average Consumption (kWh)
0.38
0.36
0.34
0.32
0.30
0.28
0.26
0.24
−4
−2
0
2
4
6
8
10
12
Temperature (C°)

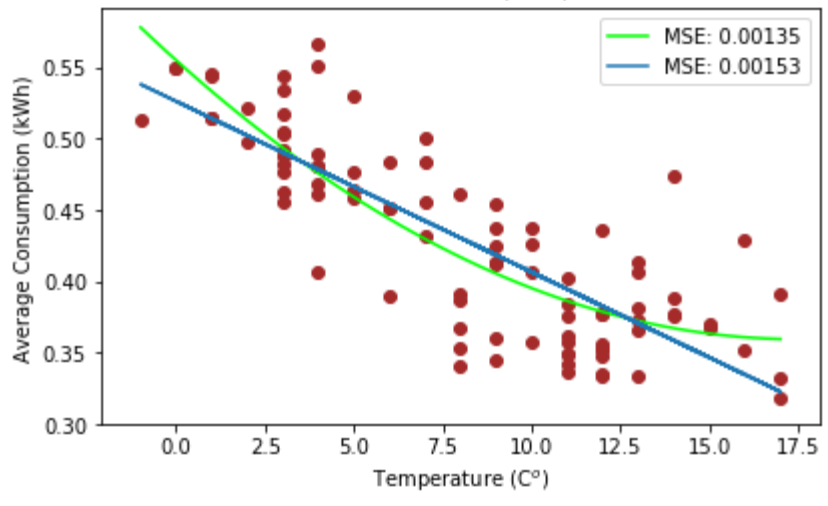

Time: 07:00 (non-TOU)
MSE: 0.0015
MSE: 0.0015
Average Consumption (kWh)
0.500
0.475
0.450
0.425
0.400
0.375
0.350
0.325
0.300
−2
0
2
4
6
8
10
12
14
Temperature (C°)

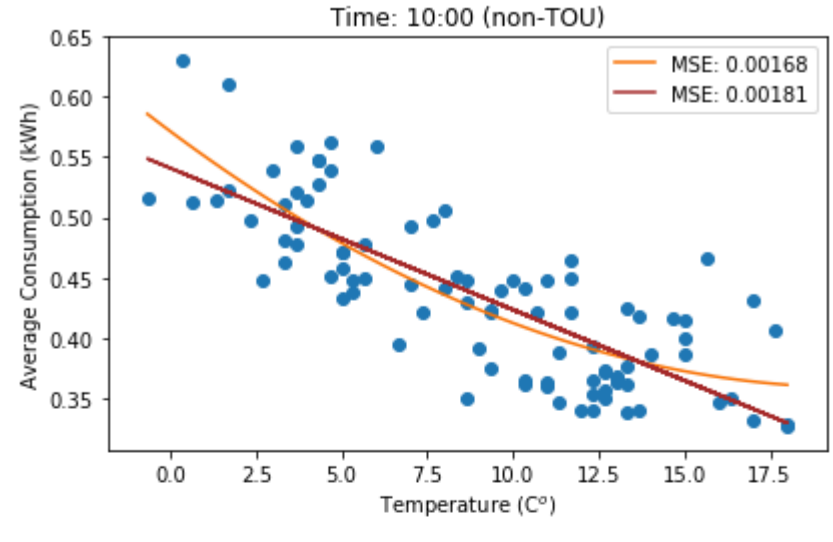

Time: 07:00 (TOU)
MSE: 0.00141
MSE: 0.00141
Average Consumption (kWh)
0.475
0.450
0.425
0.400
0.375
0.350
0.325
0.300
−2
0
2
4
6
8
10
12
14
Temperature (C°)

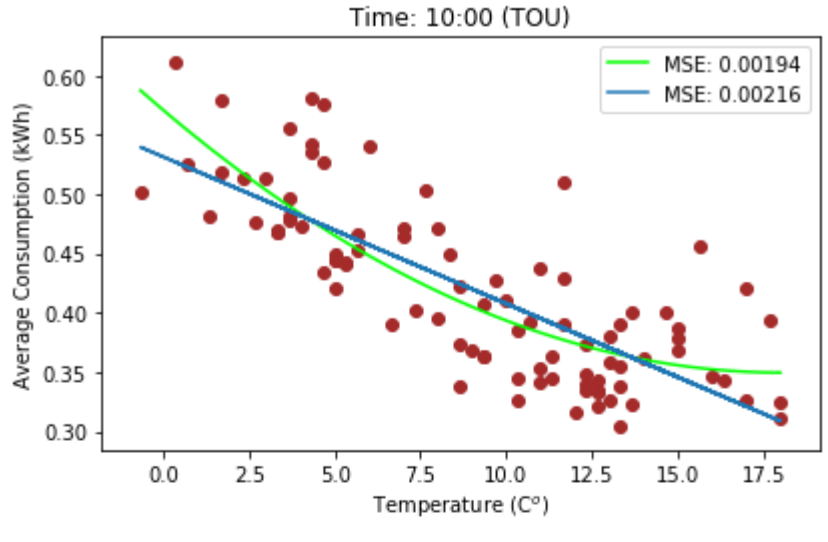

Time: 08:00 (non-TOU)
MSE: 0.00082
MSE: 0.00085
Average Consumption (kWh)
0.55
0.50
0.45
0.40
0.35
−2.5
0.0
2.5
5.0
7.5
10.0
12.5
15.0
Temperature (C°)


Time: 08:00 (TOU)
MSE: 0.00108
MSE: 0.00111
Average Consumption (kWh)
0.55
0.50
0.45
0.40
0.35
−2.5
0.0
2.5
5.0
7.5
10.0
12.5
15.0
Temperature (C°)


Time: 09:00 (non-TOU)
MSE: 0.00089
MSE: 0.00098
Average Consumption (kWh)
0.55
0.50
0.45
0.40
0.35
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Temperature (C°)

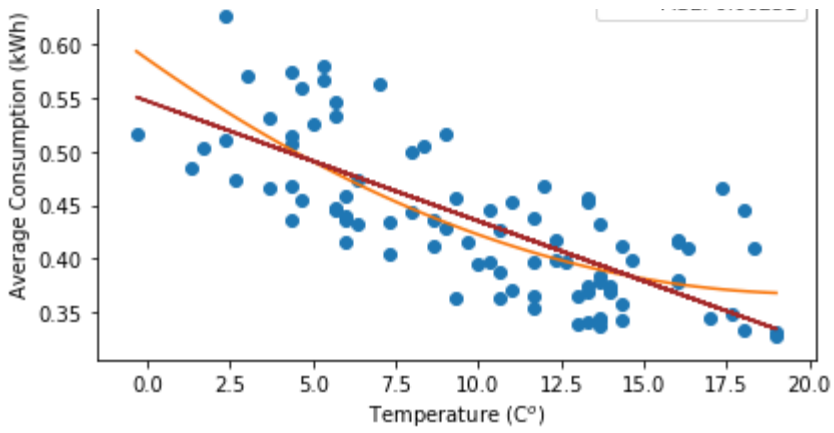

Time: 09:00 (TOU)
MSE: 0.00135
MSE: 0.00153
Average Consumption (kWh)
0.55
0.50
0.45
0.40
0.35
0.30
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Temperature (C°)

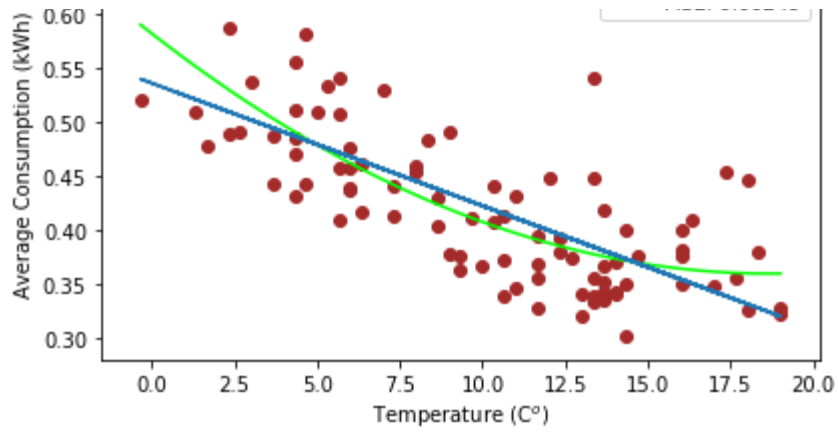

Time: 10:00 (non-TOU)
MSE: 0.00168
MSE: 0.00181
Average Consumption (kWh)
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Temperature (C°)

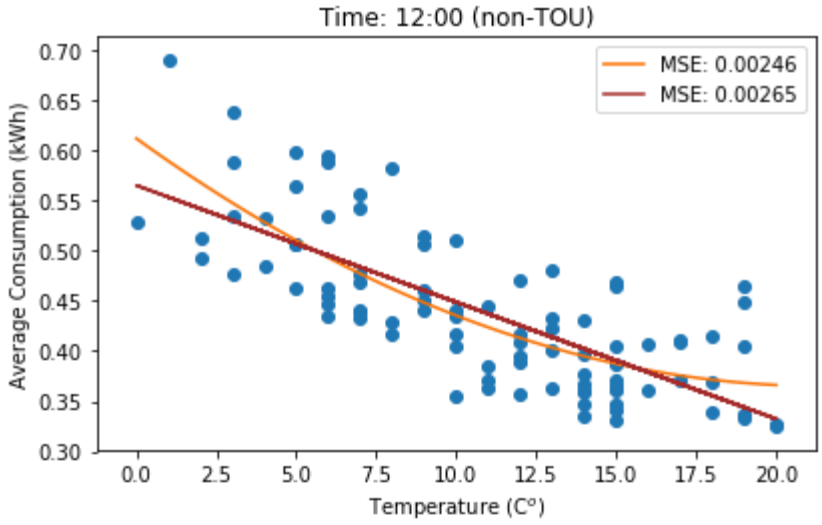

Time: 10:00 (TOU)
MSE: 0.00194
MSE: 0.00216
Average Consumption (kWh)
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Temperature (C°)

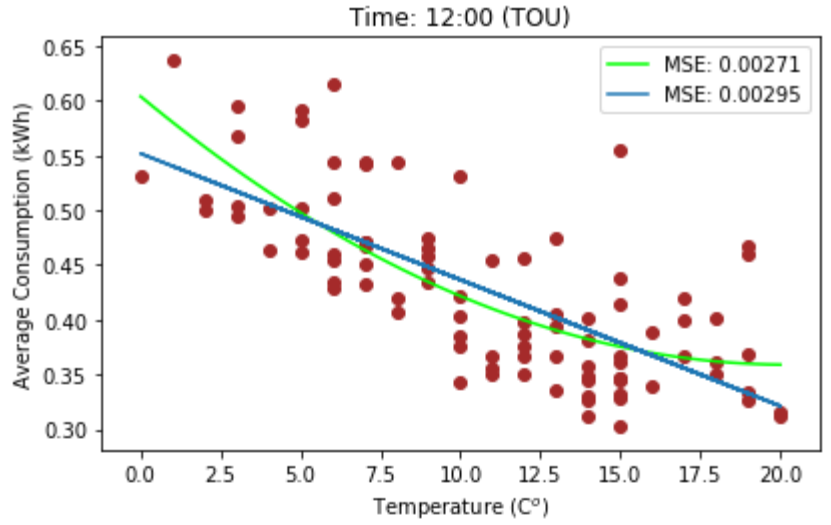
Time: 11:00 (non-TOU)
MSE: 0.00215
MSE: 0.00231
0.65

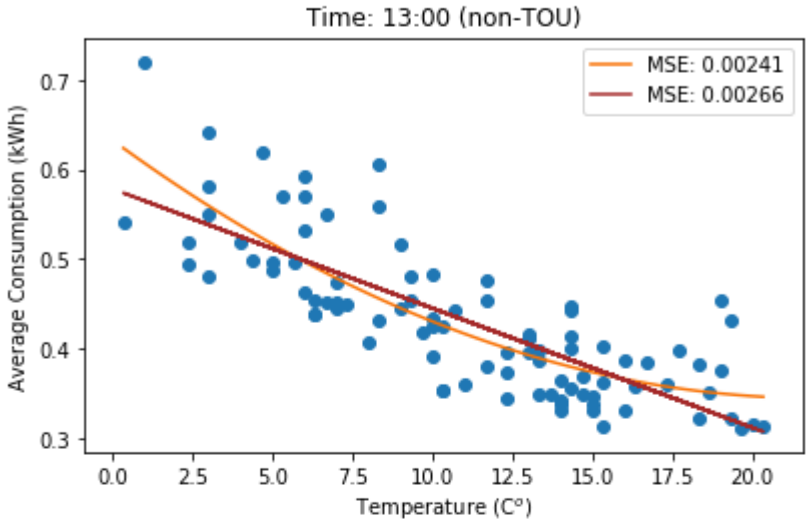
Time: 11:00 (TOU)
MSE: 0.00226
MSE: 0.00249
0.65

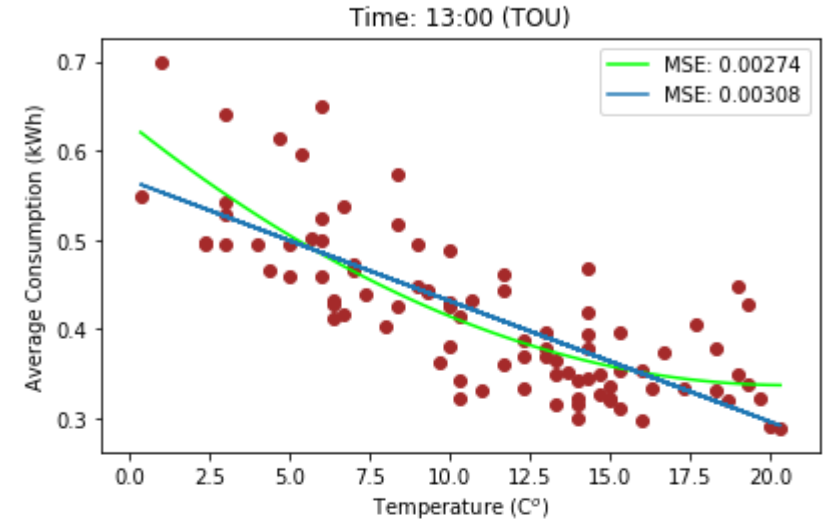

Average Consumption (kWh)
Temperature (C°)

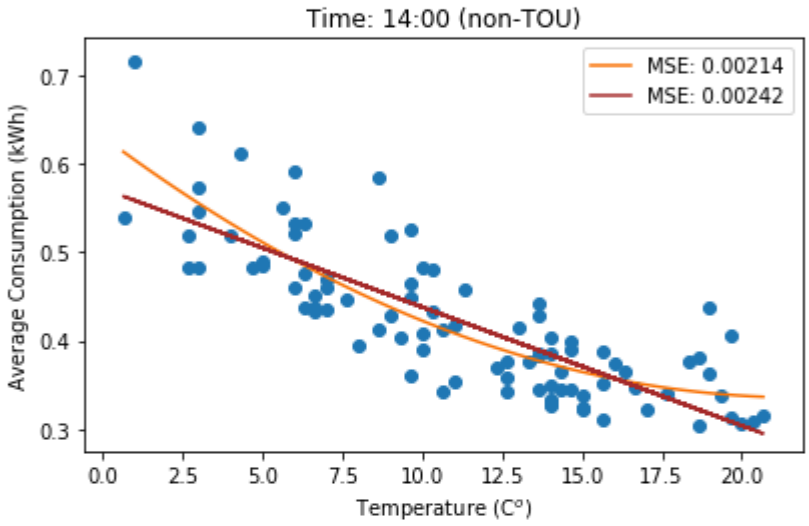

Average Consumption (kWh)
Temperature (C°)

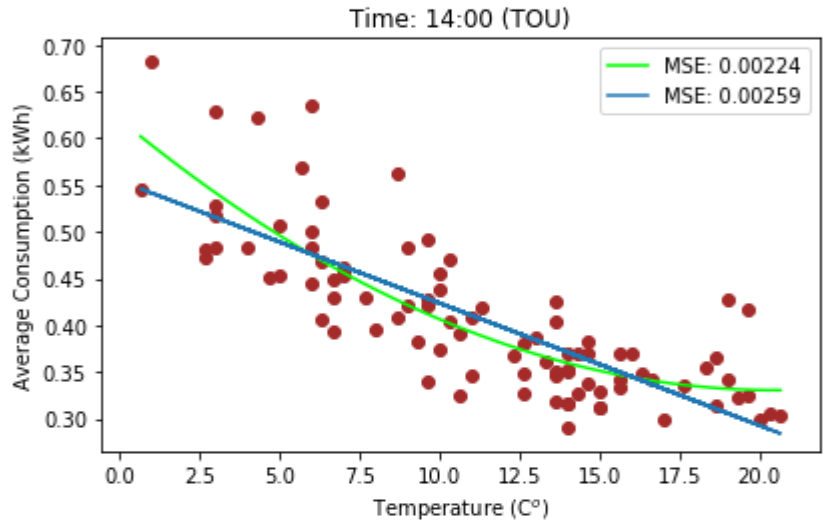

Time: 12:00 (non-TOU)
MSE: 0.00246
MSE: 0.00265
Average Consumption (kWh)
Temperature (C°)

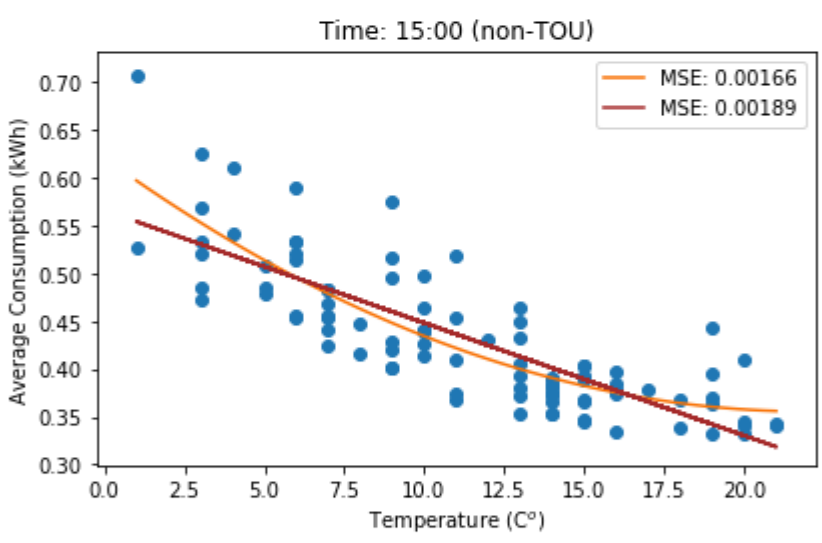

Time: 12:00 (TOU)
MSE: 0.00271
MSE: 0.00295
Average Consumption (kWh)
Temperature (C°)

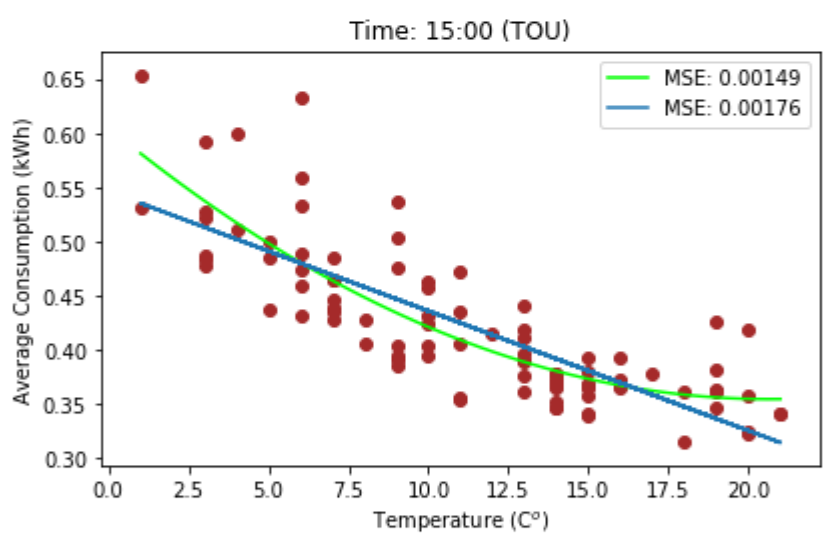

Time: 13:00 (non-TOU)
MSE: 0.00241
MSE: 0.00266
Average Consumption (kWh)
Temperature (C°)

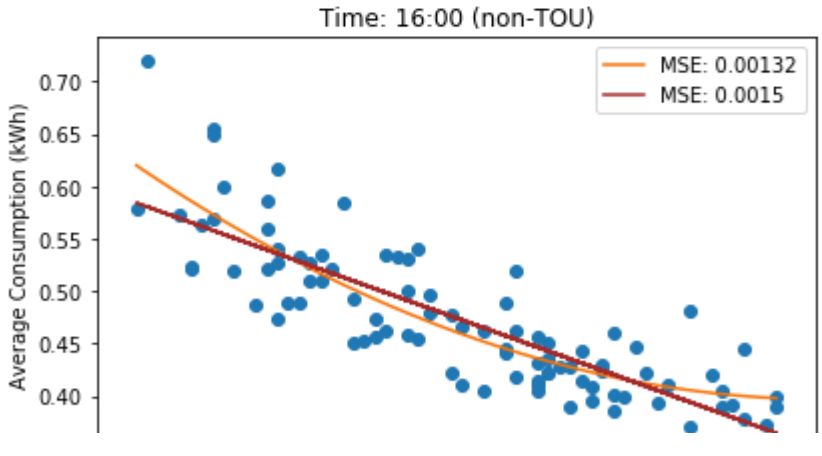

Time: 13:00 (TOU)
MSE: 0.00274
MSE: 0.00308
Average Consumption (kWh)
Temperature (C°)

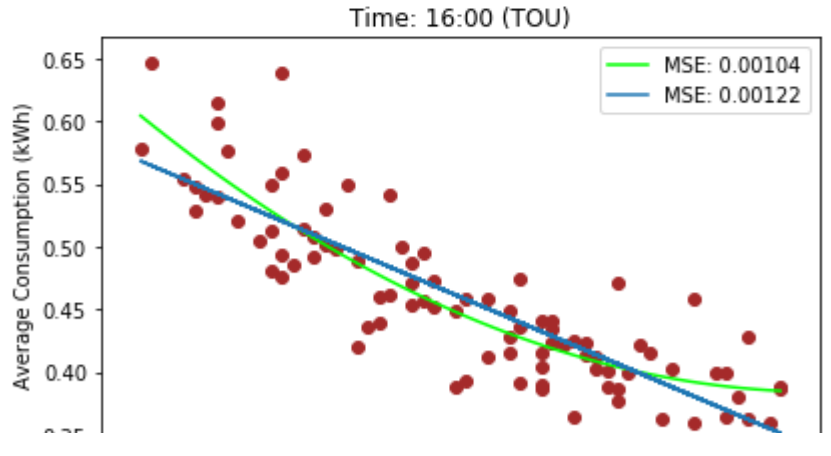

Time: 14:00 (non-TOU)
MSE: 0.00214
MSE: 0.00242
Average Consumption (kWh)
Temperature (C°)

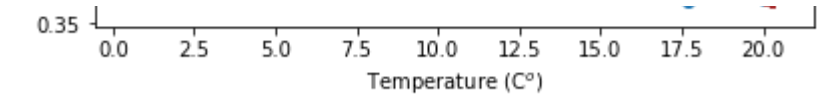

Time: 14:00 (TOU)
MSE: 0.00224
MSE: 0.00259
Average Consumption (kWh)
Temperature (C°)

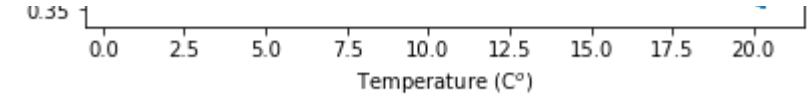

Time: 15:00 (non-TOU)
MSE: 0.00166
MSE: 0.00189
Average Consumption (kWh)
Temperature (C°)

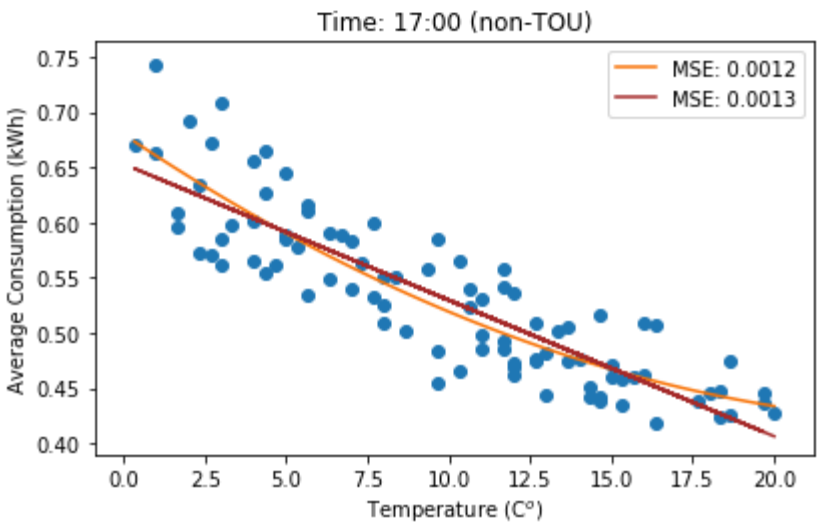

Time: 15:00 (TOU)
MSE: 0.00149
MSE: 0.00176
Average Consumption (kWh)
Temperature (C°)

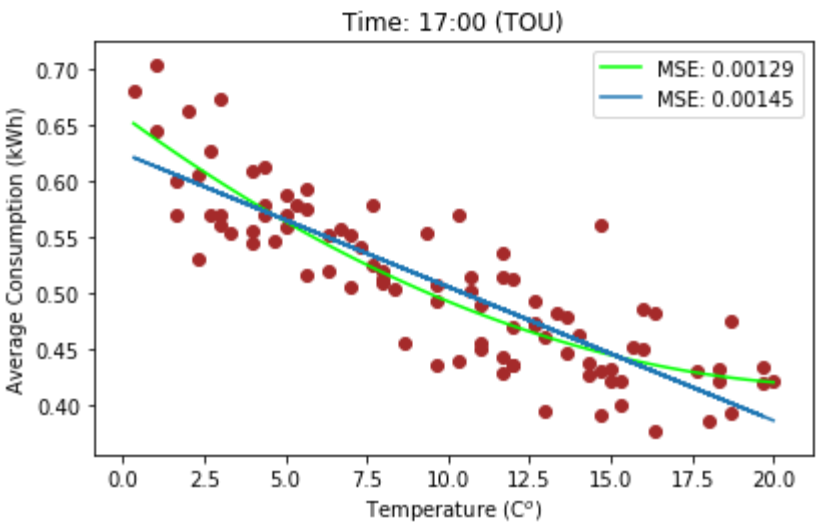

Time: 16:00 (non-TOU)
MSE: 0.00132
MSE: 0.0015
Average Consumption (kWh)

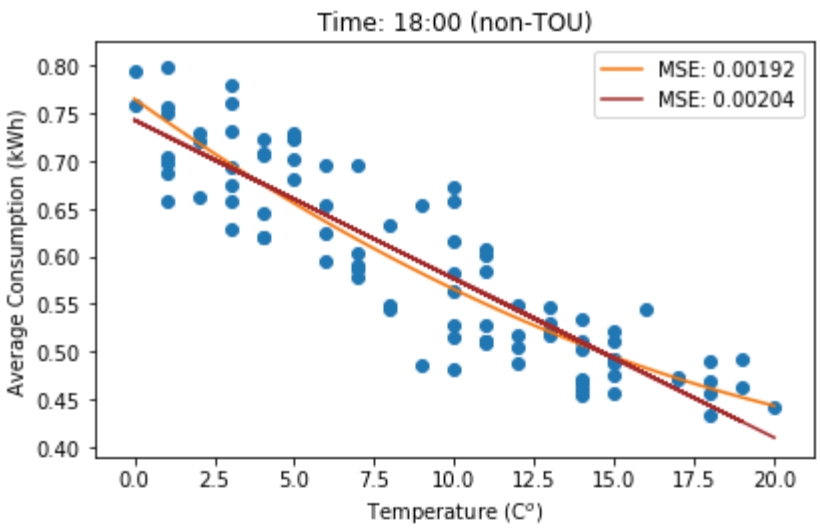

Time: 16:00 (TOU)
MSE: 0.00104
MSE: 0.00122
Average Consumption (kWh)

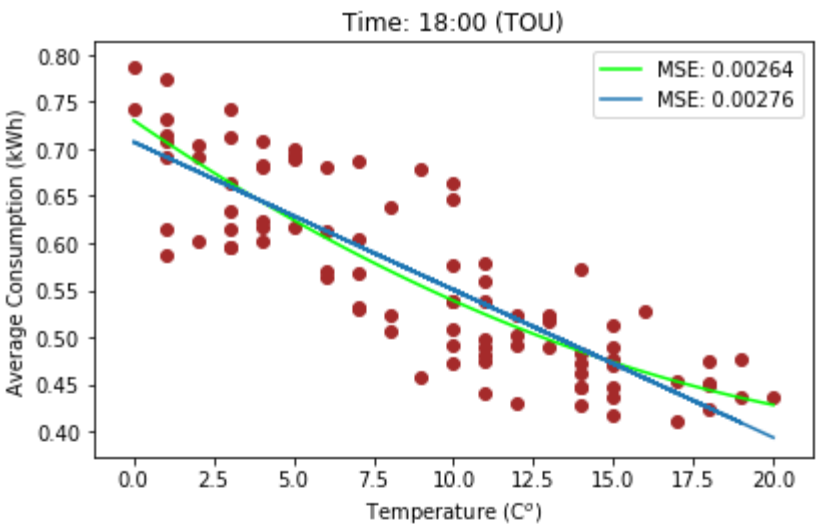

0.35
0.0 2.5 5.0 7.5 10.0 12.5 15.0 17.5 20.0
Temperature (C°)

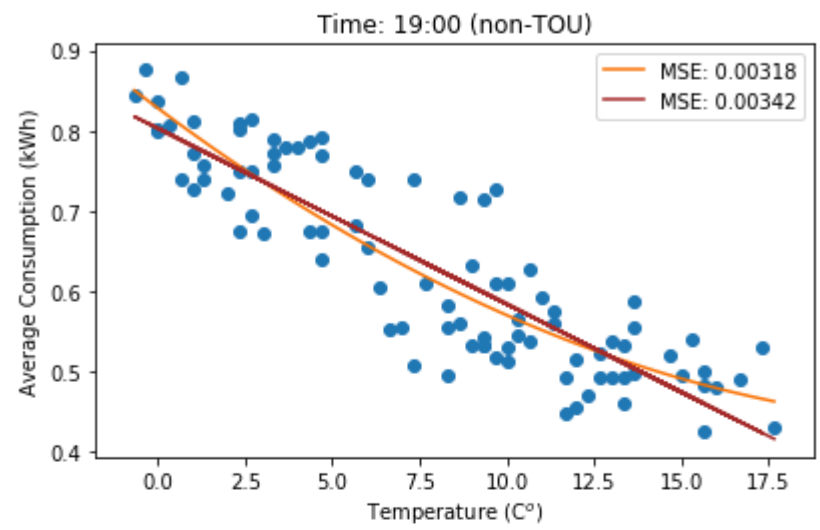

0.0 2.5 5.0 7.5 10.0 12.5 15.0 17.5 20.0
Temperature (C°)

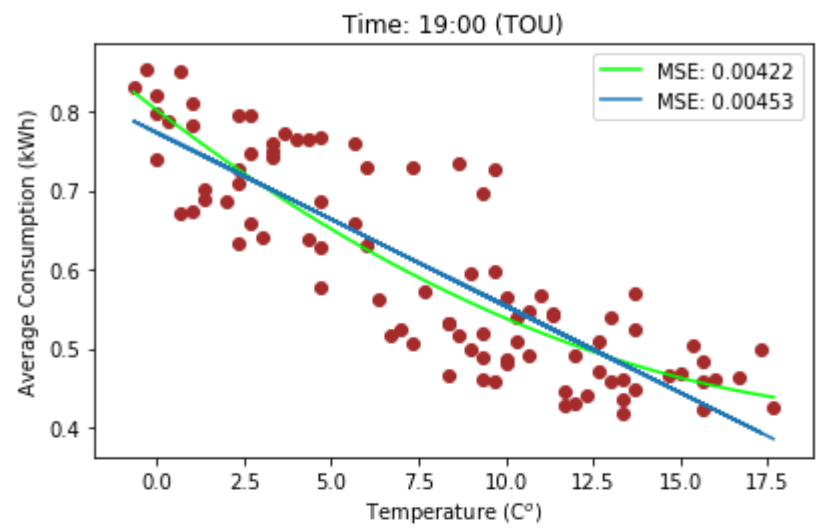

Time: 17:00 (non-TOU)
MSE: 0.0012
MSE: 0.0013
Average Consumption (kWh)
0.75 0.70 0.65 0.60 0.55 0.50 0.45 0.40
0.0 2.5 5.0 7.5 10.0 12.5 15.0 17.5 20.0
Temperature (C°)

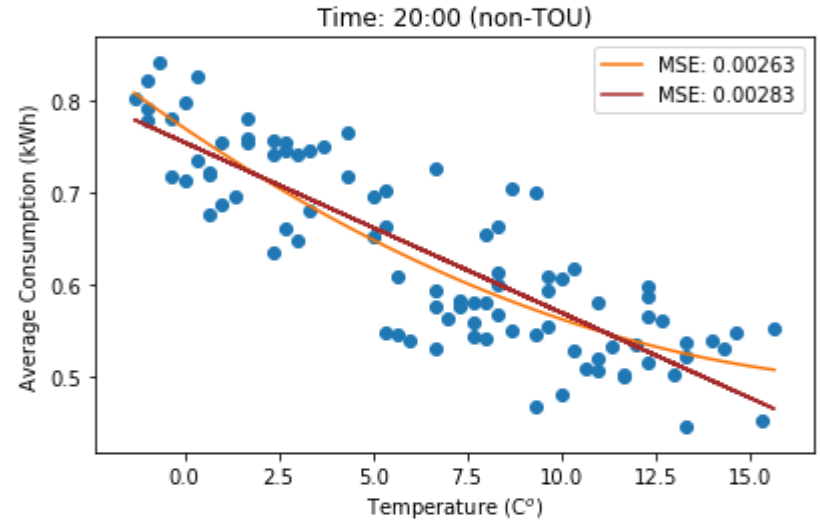

Time: 17:00 (TOU)
MSE: 0.00129
MSE: 0.00145
Average Consumption (kWh)
0.70 0.65 0.60 0.55 0.50 0.45 0.40
0.0 2.5 5.0 7.5 10.0 12.5 15.0 17.5 20.0
Temperature (C°)

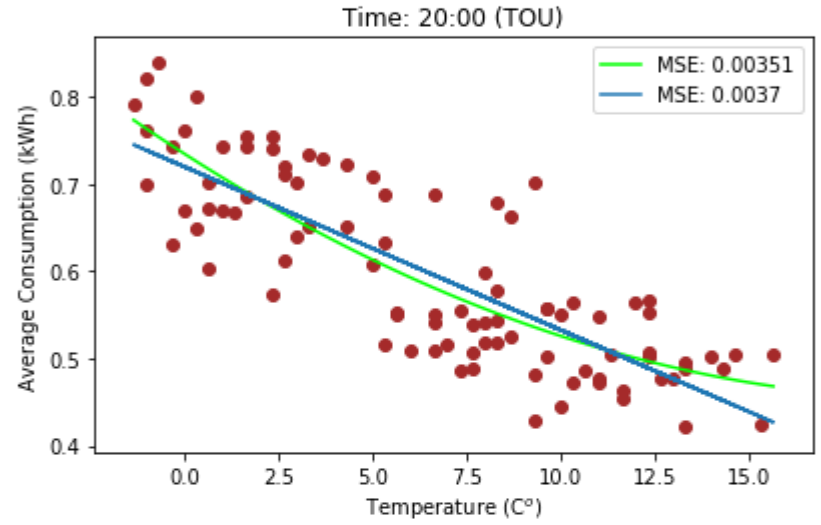

Time: 18:00 (non-TOU)
MSE: 0.00192
MSE: 0.00204
Average Consumption (kWh)
0.80 0.75 0.70 0.65 0.60 0.55 0.50 0.45 0.40
0.0 2.5 5.0 7.5 10.0 12.5 15.0 17.5 20.0
Temperature (C°)

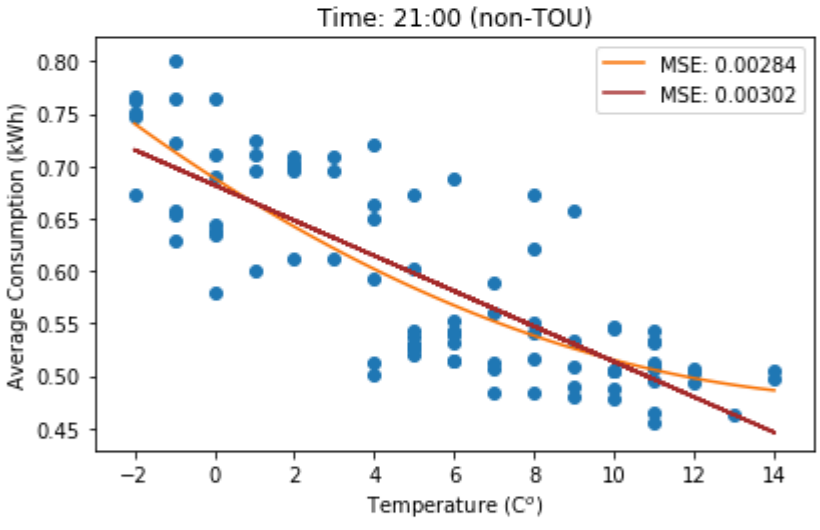

Time: 18:00 (TOU)
MSE: 0.00264
MSE: 0.00276
Average Consumption (kWh)
0.80 0.75 0.70 0.65 0.60 0.55 0.50 0.45 0.40
0.0 2.5 5.0 7.5 10.0 12.5 15.0 17.5 20.0
Temperature (C°)

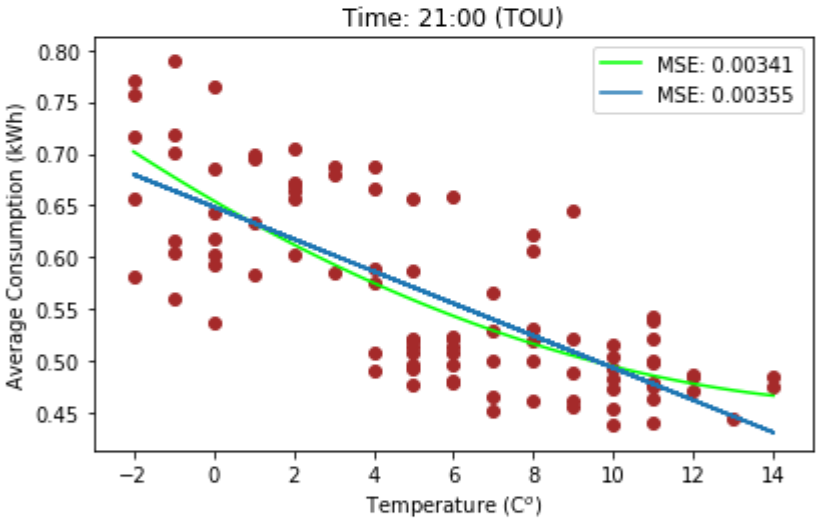

Time: 19:00 (non-TOU)
MSE: 0.00318
MSE: 0.00342
Average Consumption (kWh)
0.9 0.8 0.7 0.6 0.5 0.4
0.0 2.5 5.0 7.5 10.0 12.5 15.0 17.5
Temperature (C°)


Time: 19:00 (TOU)
MSE: 0.00422
MSE: 0.00453
Average Consumption (kWh)
0.8 0.7 0.6 0.5 0.4
0.0 2.5 5.0 7.5 10.0 12.5 15.0 17.5
Temperature (C°)


Time: 20:00 (non-TOU)
MSE: 0.00263
MSE: 0.00283
Average Consumption (kWh)
0.8 0.7 0.6 0.5
0.0 2.5 5.0 7.5 10.0 12.5 15.0
Temperature (C°)

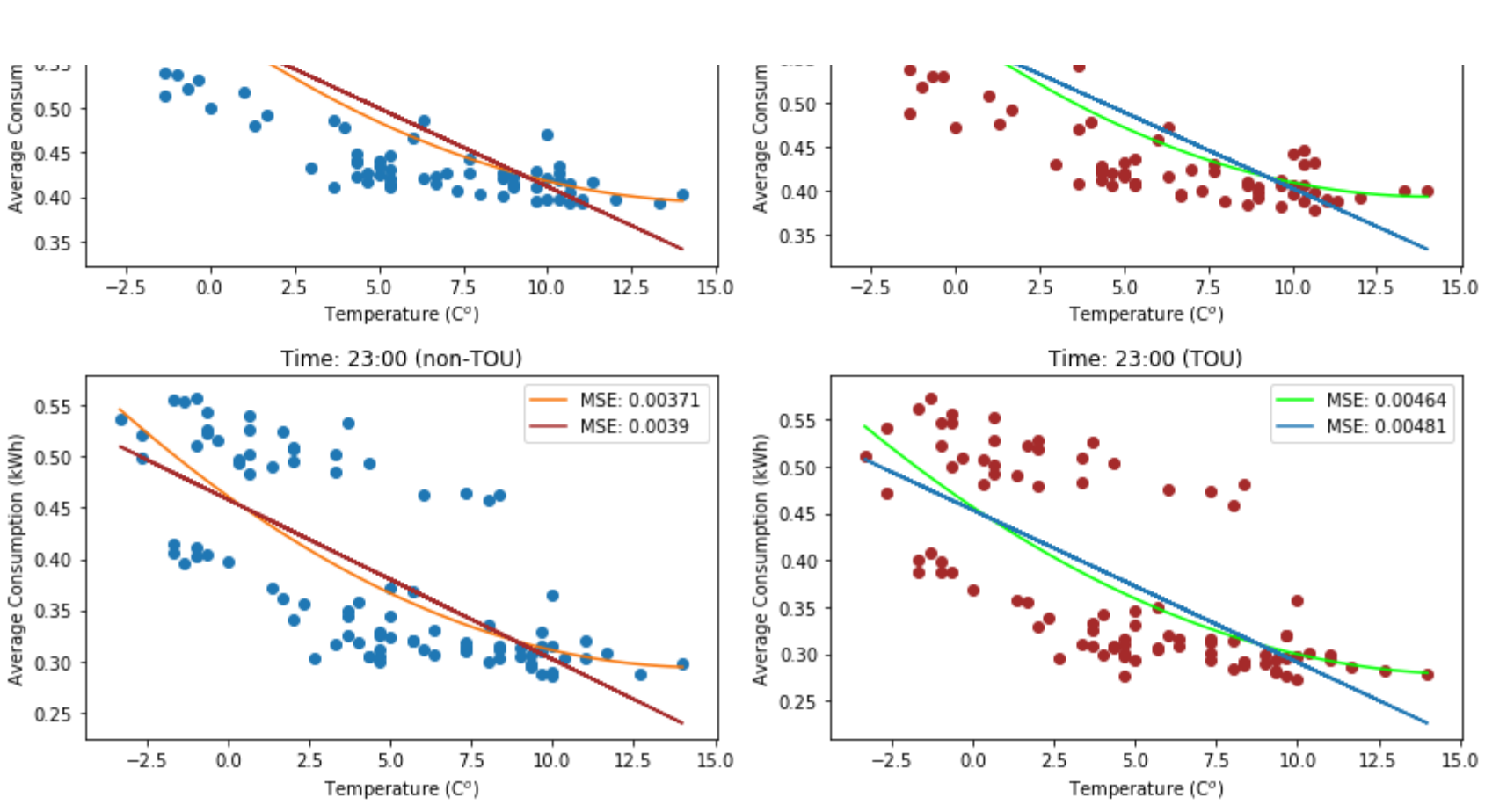

Time: 20:00 (TOU)
MSE: 0.00351
MSE: 0.0037
Average Consumption (kWh)
0.8 0.7 0.6 0.5 0.4
0.0 2.5 5.0 7.5 10.0 12.5 15.0
Temperature (C°)


Time: 21:00 (non-TOU)
MSE: 0.00284
MSE: 0.00302
Average Consumption (kWh)
0.80 0.75 0.70 0.65 0.60 0.55 0.50 0.45
−2 0 2 4 6 8 10 12 14
Temperature (C°)

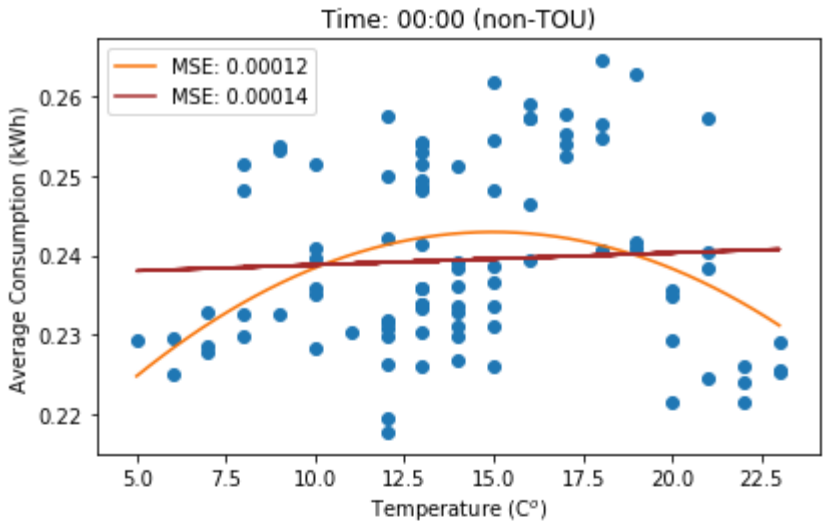

Time: 21:00 (TOU)
MSE: 0.00341
MSE: 0.00355
Average Consumption (kWh)
0.80 0.75 0.70 0.65 0.60 0.55 0.50 0.45
−2 0 2 4 6 8 10 12 14
Temperature (C°)

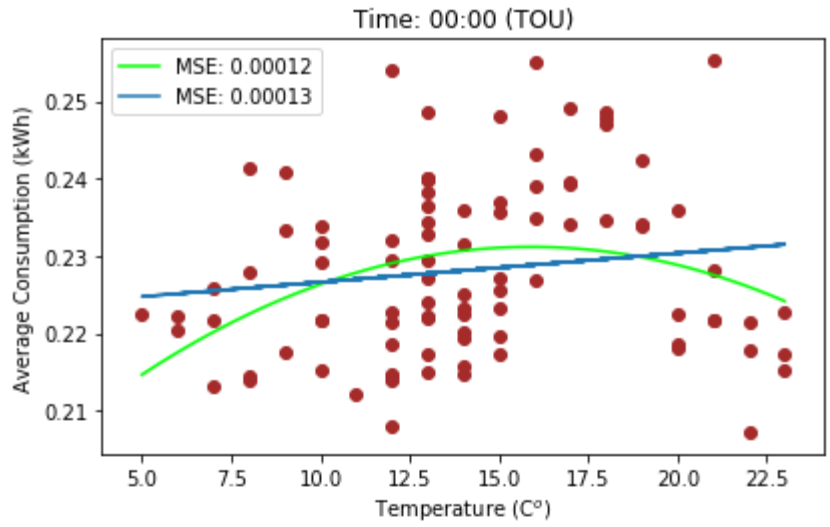
Time: 22:00 (non-TOU)
MSE: 0.00379
MSE: 0.00402
0.70 0.65 0.60 0.55
0.0 2.5 5.0 7.5 10.0 12.5 15.0
Temperature (C°)

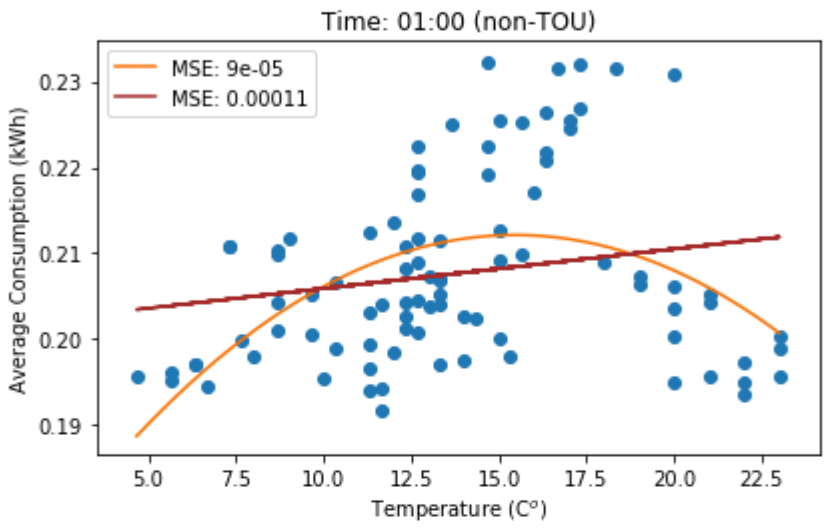
Time: 22:00 (TOU)
MSE: 0.00418
MSE: 0.00445
0.70 0.65 0.60 0.55
0.0 2.5 5.0 7.5 10.0 12.5 15.0
Temperature (C°)

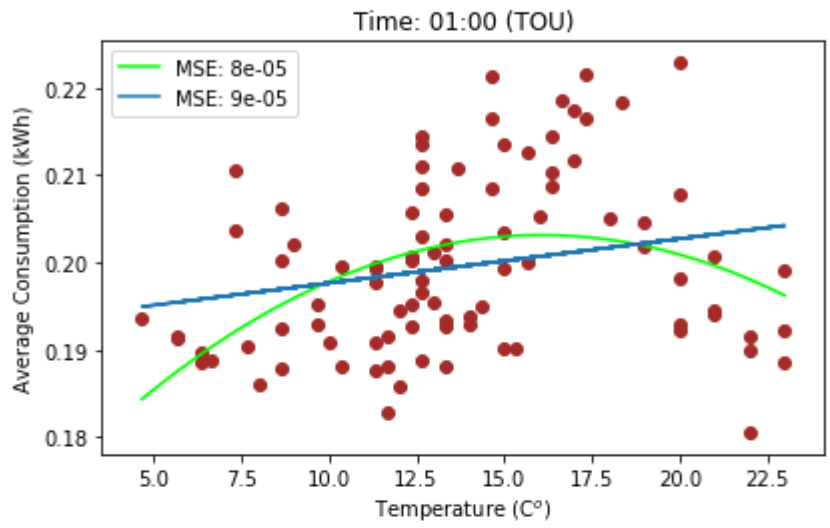

Average Consum
Temperature (C°)
Average Consum
Temperature (C°)
Time: 23:00 (non-TOU)
MSE: 0.00371
MSE: 0.0039
Average Consumption (kWh)
Temperature (C°)
Time: 23:00 (TOU)
MSE: 0.00464
MSE: 0.00481
Average Consumption (kWh)
Temperature (C°)

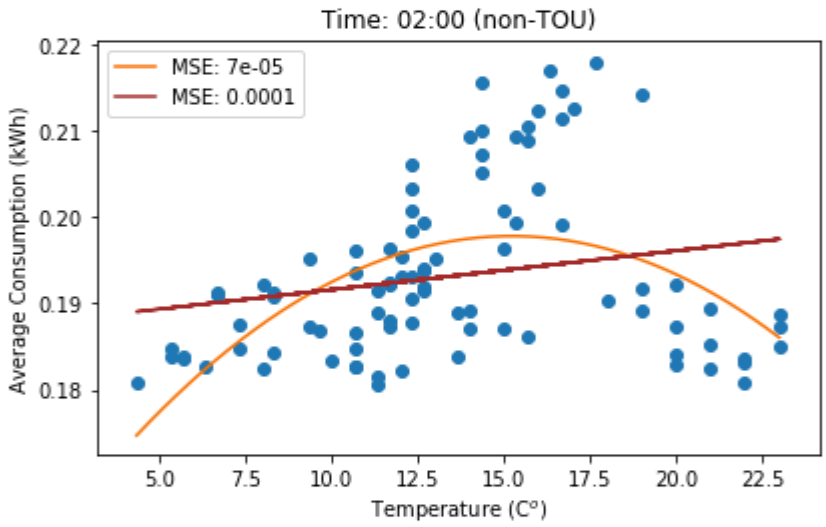
In [20]:
```
# Summer scatterplot of average consumption (over all consumers at same time period) vs. temp
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
# non-TOU vs TOU in terms of showing the elements in Average column such that they are hourly energy consumpt
ion v.s. temperature over 365 days
fig_all = plt.figure(figsize = (12,90))
ax_Ntou = [] # store subplot objects
ax_tou = [] # store subplot objects
lm = LinearRegression()
for i in range(24):
    ax_Ntou.append(fig_all.add_subplot(24, 2, 2 * i + 1))
    if i <= 9:
        x = df_wea1h_summer.TempC[df_wea1h_summer.GMT.str.contains('0' + str(i) + ':00:00')].values
        y = df_Ntou1h_summer.Average[df_Ntou1h_summer.GMT.str.contains('0' + str(i) + ':00:00')].values
        xp = np.linspace(min(x), max(x), 50) # polynomial regression
        z = np.polyfit(x, y, 2)
        p = np.poly1d(z)
        ax_Ntou[-1].plot(xp, p(xp), color = 'xkcd:orange', label = 'MSE: ' + str(round(mean_squared_error(y,
p(x)), 5)))
        x = x.reshape(-1, 1)
        y = y.reshape(-1, 1)
        lm.fit(x, y)
        ax_Ntou[-1].plot(x, lm.predict(x), color = 'brown', label = 'MSE: ' + str(round(mean_squared_error(y,
lm.predict(x)), 5)))
        ax_Ntou[-1].scatter(x, y)
        ax_Ntou[-1].set_title('Time: 0' + str(i) + ':00 (non-TOU)')
        ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_Ntou[-1].set_ylabel('Average Consumption (kWh)')
        ax_Ntou[-1].legend()
    else:
        x = df_wea1h_summer.TempC[df_wea1h_summer.GMT.str.contains(str(i) + ':00:00')].values
        y = df_Ntou1h_summer.Average[df_Ntou1h_summer.GMT.str.contains(str(i) + ':00:00')].values
        xp = np.linspace(min(x), max(x), 50) # polynomial regression
        z = np.polyfit(x, y, 2)
        p = np.poly1d(z)
        ax_Ntou[-1].plot(xp, p(xp), color = 'xkcd:orange', label = 'MSE: ' + str(round(mean_squared_error(y,
p(x)), 5)))
        x = x.reshape(-1, 1)
        y = y.reshape(-1, 1)
        lm.fit(x, y)
        ax_Ntou[-1].plot(x, lm.predict(x), color = 'brown', label = 'MSE: ' + str(round(mean_squared_error(y,
lm.predict(x)), 5)))
        ax_Ntou[-1].scatter(x, y)
        ax_Ntou[-1].set_title('Time: ' + str(i) + ':00 (non-TOU)')
        ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_Ntou[-1].set_ylabel('Average Consumption (kWh)')
        ax_Ntou[-1].legend()
for i in range(24):
    ax_tou.append(fig_all.add_subplot(24, 2, 2 * i + 2))
    if i <= 9:
        x = df_wea1h_summer.TempC[df_wea1h_summer.GMT.str.contains('0' + str(i) + ':00:00')].values
        y = df_tou1h_summer.Average[df_tou1h_summer.GMT.str.contains('0' + str(i) + ':00:00')].values
        xp = np.linspace(min(x), max(x), 50) # polynomial regression
        z = np.polyfit(x, y, 2)
        p = np.poly1d(z)
        ax_tou[-1].plot(xp, p(xp), color = 'lime', label = 'MSE: ' + str(round(mean_squared_error(y, p(x)), 5
)))
        x = x.reshape(-1, 1)
        y = y.reshape(-1, 1)
        lm.fit(x, y)
        ax_tou[-1].plot(x, lm.predict(x), label = 'MSE: ' + str(round(mean_squared_error(y, lm.predict(x)), 5
)))
        ax_tou[-1].scatter(x, y, color = 'brown')
        ax_tou[-1].set_title('Time: 0' + str(i) + ':00 (TOU)')
        ax_tou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_tou[-1].set_ylabel('Average Consumption (kWh)')
        ax_tou[-1].legend()
    else:
        x = df_wea1h_summer.TempC[df_wea1h_summer.GMT.str.contains(str(i) + ':00:00')].values
        y = df_tou1h_summer.Average[df_tou1h_summer.GMT.str.contains(str(i) + ':00:00')].values
        xp = np.linspace(min(x), max(x), 50) # polynomial regression
        z = np.polyfit(x, y, 2)
        p = np.poly1d(z)
        ax_tou[-1].plot(xp, p(xp), color = 'lime', label = 'MSE: ' + str(round(mean_squared_error(y, p(x)), 5
)))
        x = x.reshape(-1, 1)
        y = y.reshape(-1, 1)
        lm.fit(x, y)
        ax_tou[-1].plot(x, lm.predict(x), label = 'MSE: ' + str(round(mean_squared_error(y, lm.predict(x)), 5
)))
        ax_tou[-1].scatter(x, y, color = 'brown')
        ax_tou[-1].set_title('Time: ' + str(i) + ':00 (TOU)')
        ax_tou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_tou[-1].set_ylabel('Average Consumption (kWh)')
        ax_tou[-1].legend()
plt.tight_layout()
```

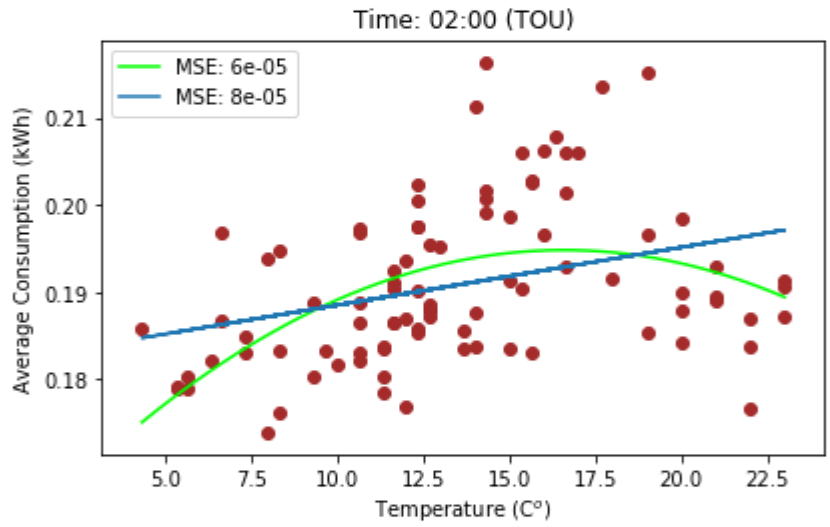

Time: 00:00 (non-TOU)
MSE: 0.00012
MSE: 0.00014
Average Consumption (kWh)
Temperature (C°)

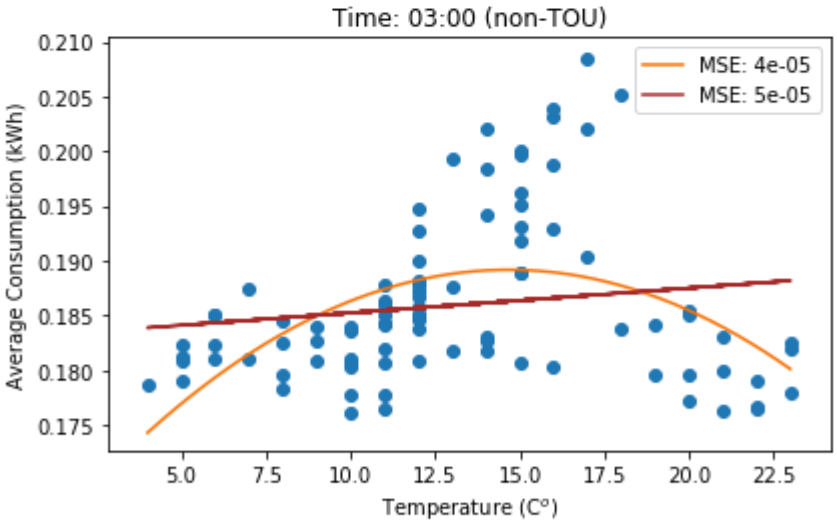

Time: 00:00 (TOU)
MSE: 0.00012
MSE: 0.00013
Average Consumption (kWh)
Temperature (C°)

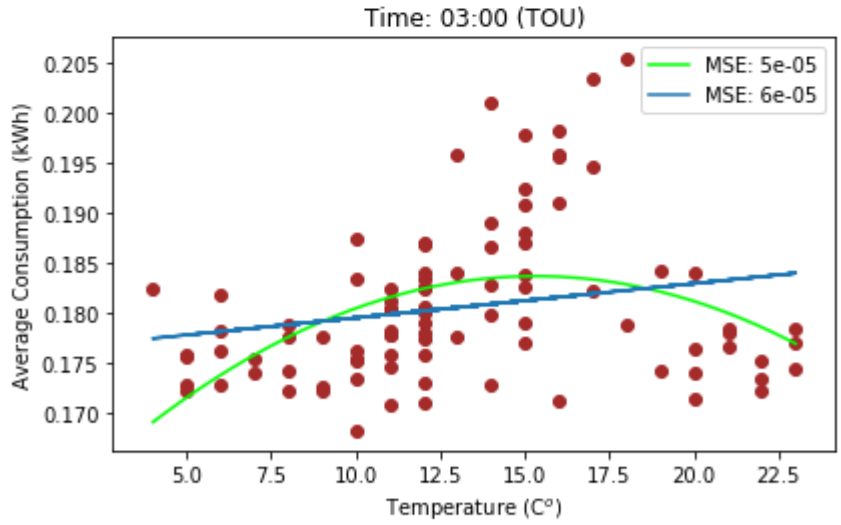

Time: 01:00 (non-TOU)
MSE: 9e-05
MSE: 0.00011
Average Consumption (kWh)
Temperature (C°)

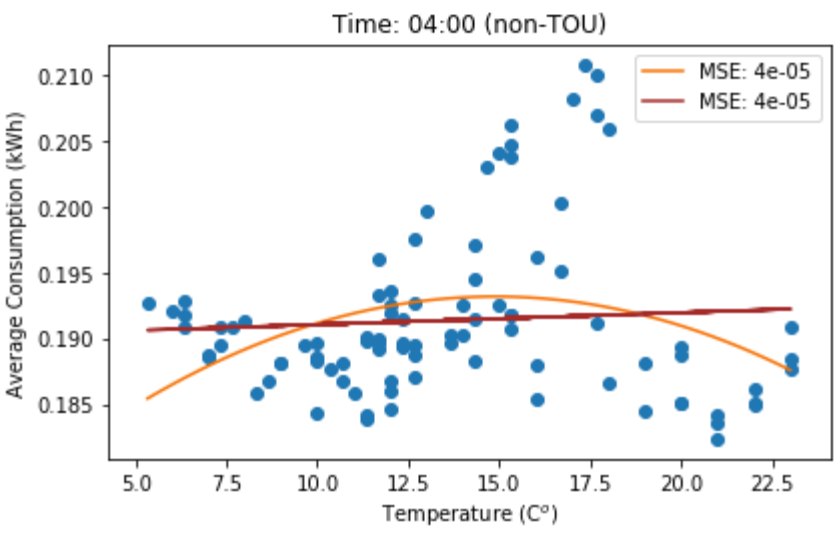

Time: 01:00 (TOU)
MSE: 8e-05
MSE: 9e-05
Average Consumption (kWh)
Temperature (C°)

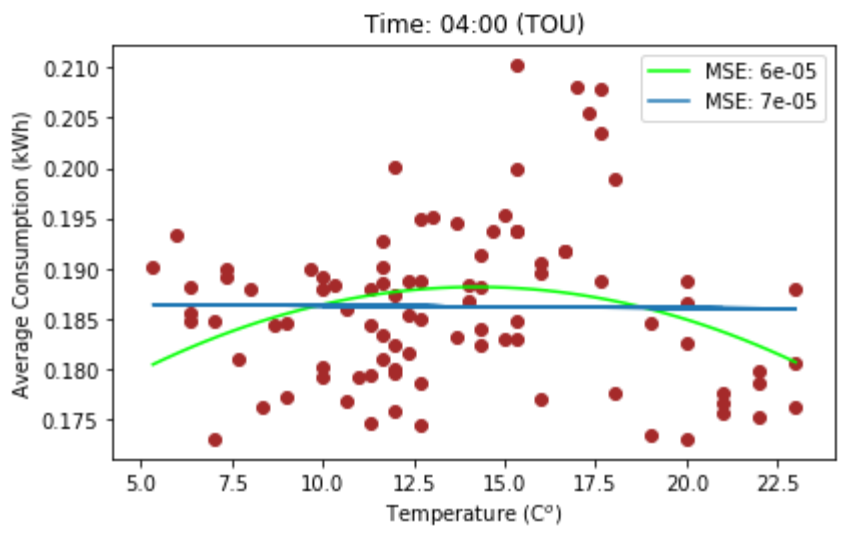

Time: 02:00 (non-TOU)
MSE: 7e-05
MSE: 0.0001
Average Consumption (kWh)
Temperature (C°)

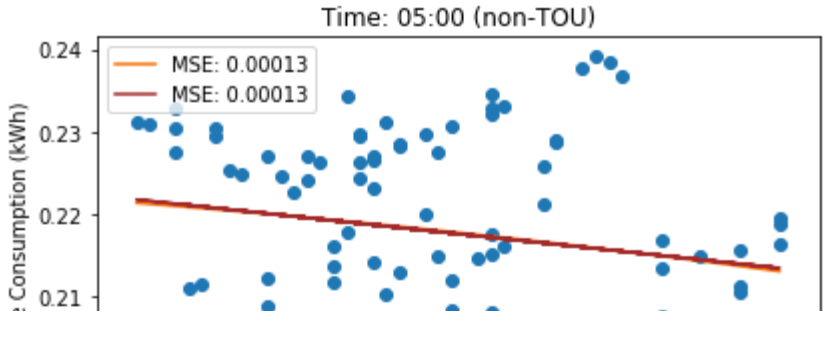

Time: 02:00 (TOU)
MSE: 6e-05
MSE: 8e-05
Average Consumption (kWh)
Temperature (C°)

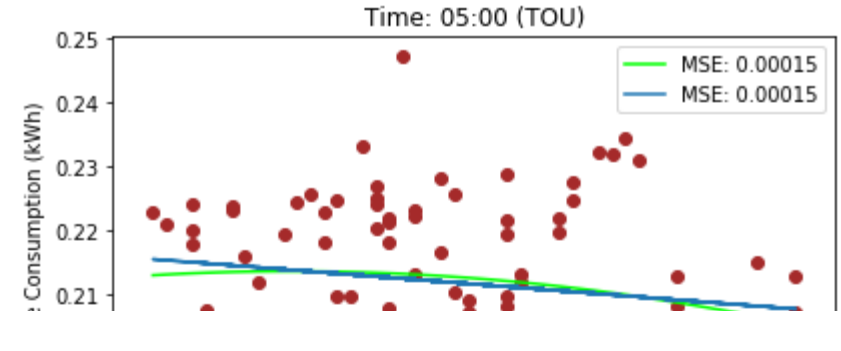

Time: 03:00 (non-TOU)
MSE: 4e-05
MSE: 5e-05
Average Consumption (kWh)
Temperature (C°)

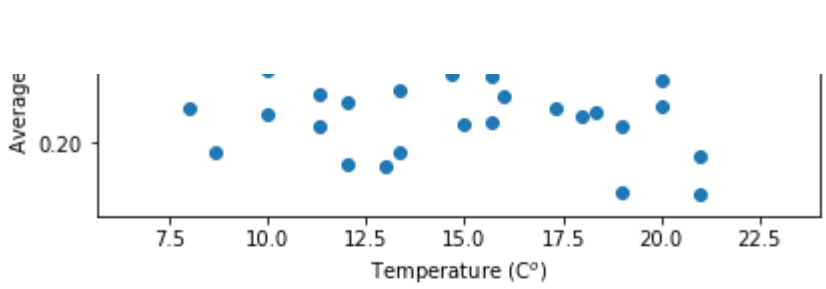

Time: 03:00 (TOU)
MSE: 5e-05
MSE: 6e-05
Average Consumption (kWh)
Temperature (C°)

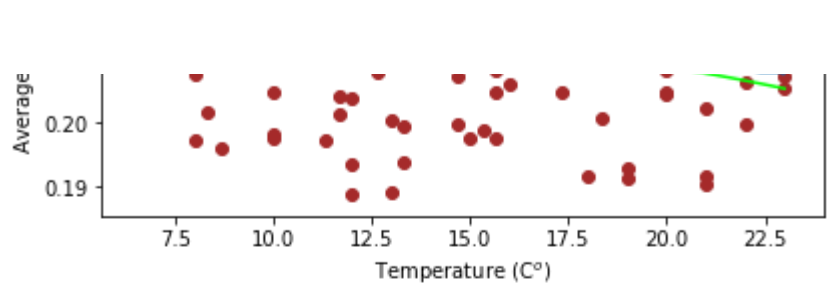

Time: 04:00 (non-TOU)
MSE: 4e-05
MSE: 4e-05
Average Consumption (kWh)
Temperature (C°)

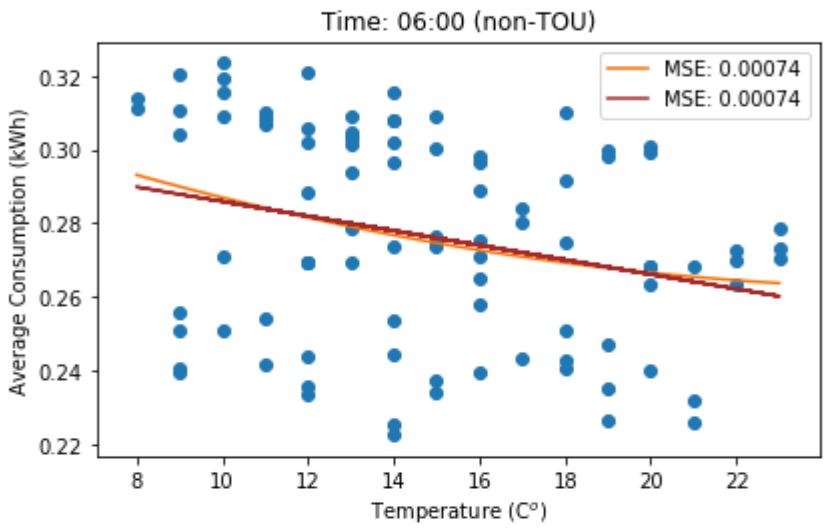

Time: 04:00 (TOU)
MSE: 6e-05
MSE: 7e-05
Average Consumption (kWh)
Temperature (C°)

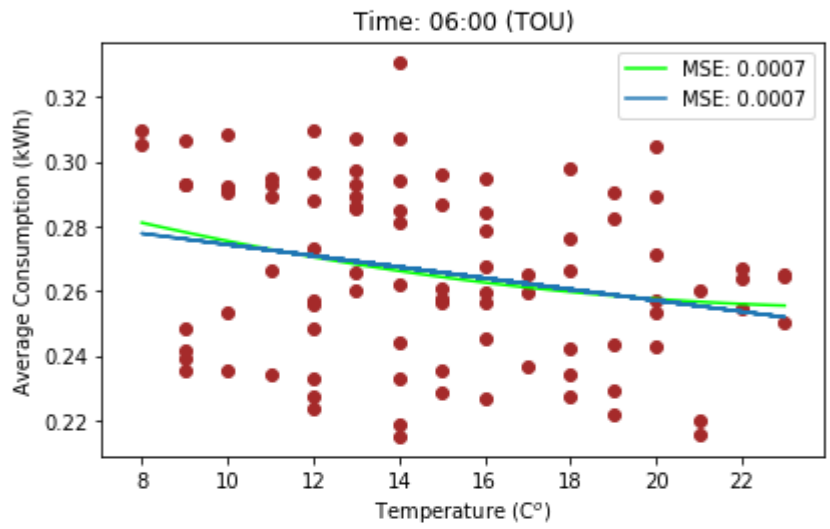

Time: 05:00 (non-TOU)
MSE: 0.00013
MSE: 0.00013
Temperature (C°)

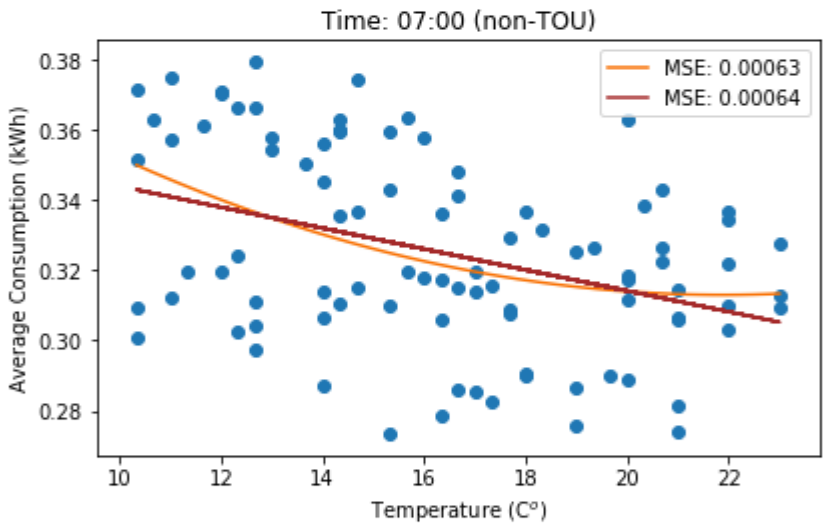

Time: 05:00 (TOU)
MSE: 0.00015
MSE: 0.00015
Temperature (C°)

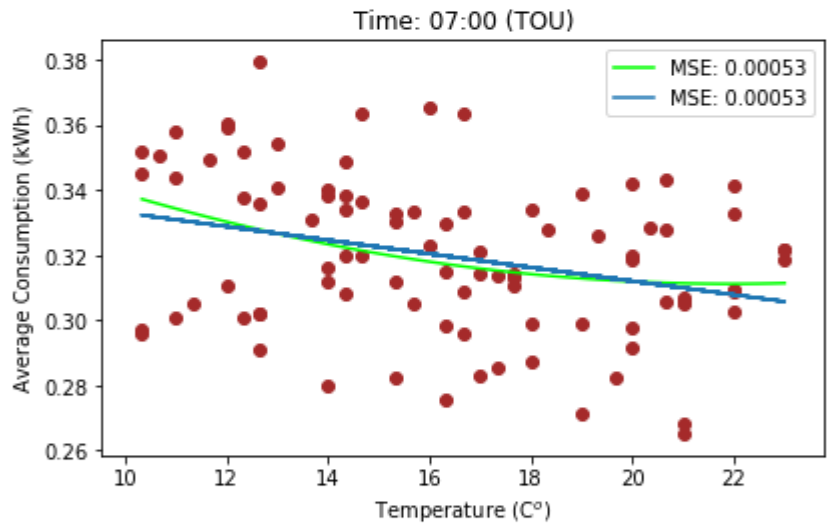

Average
0.20
7.5
10.0
12.5
15.0
17.5
20.0
22.5
Temperature (C°)

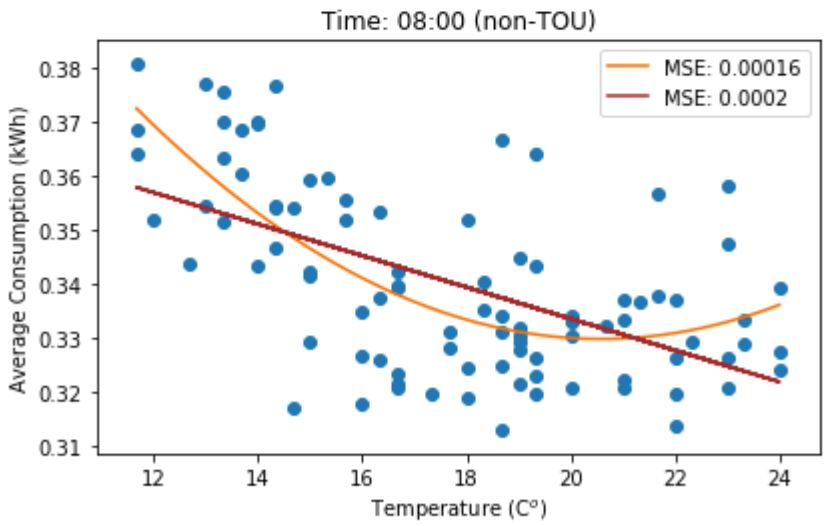

Average
0.20
0.19
7.5
10.0
12.5
15.0
17.5
20.0
22.5
Temperature (C°)

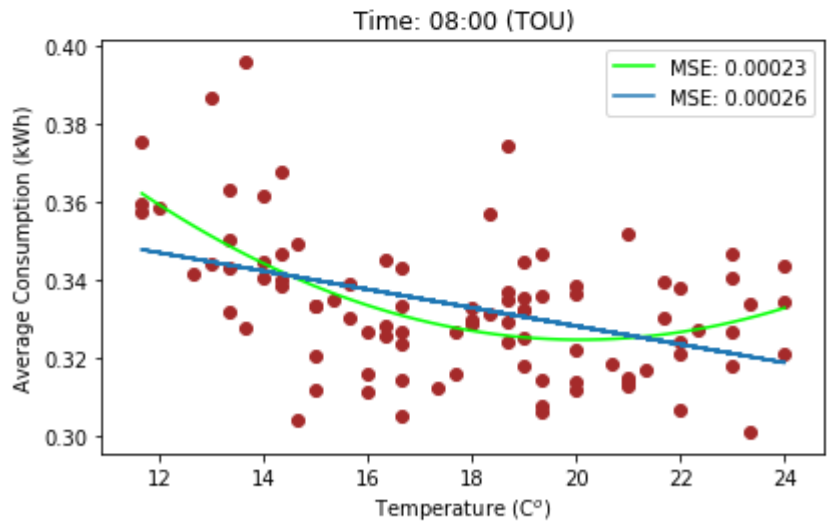

Time: 06:00 (non-TOU)
MSE: 0.00074
MSE: 0.00074
Average Consumption (kWh)
0.32
0.30
0.28
0.26
0.24
0.22
8
10
12
14
16
18
20
22
Temperature (C°)

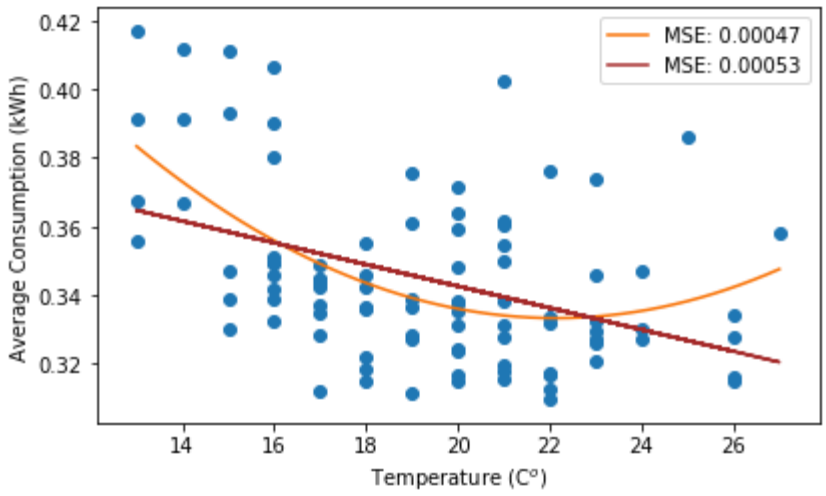

Time: 06:00 (TOU)
MSE: 0.0007
MSE: 0.0007
Average Consumption (kWh)
0.32
0.30
0.28
0.26
0.24
0.22
8
10
12
14
16
18
20
22
Temperature (C°)

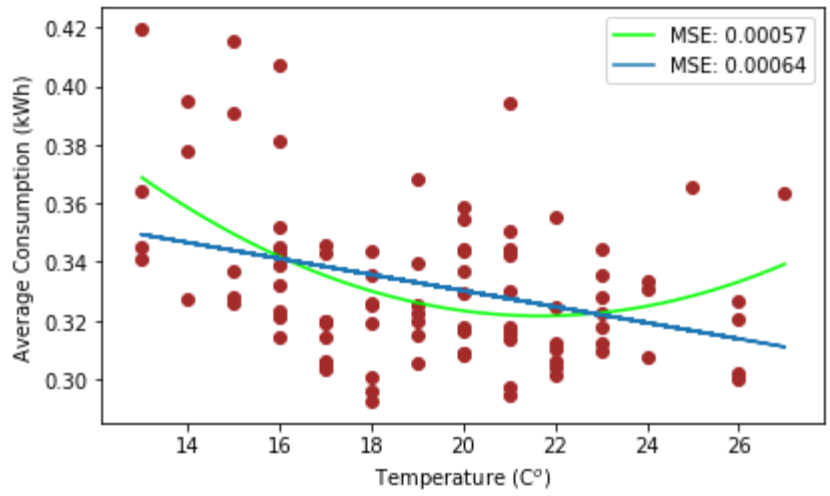

Time: 07:00 (non-TOU)
MSE: 0.00063
MSE: 0.00064
Average Consumption (kWh)
0.38
0.36
0.34
0.32
0.30
0.28
10
12
14
16
18
20
22
Temperature (C°)

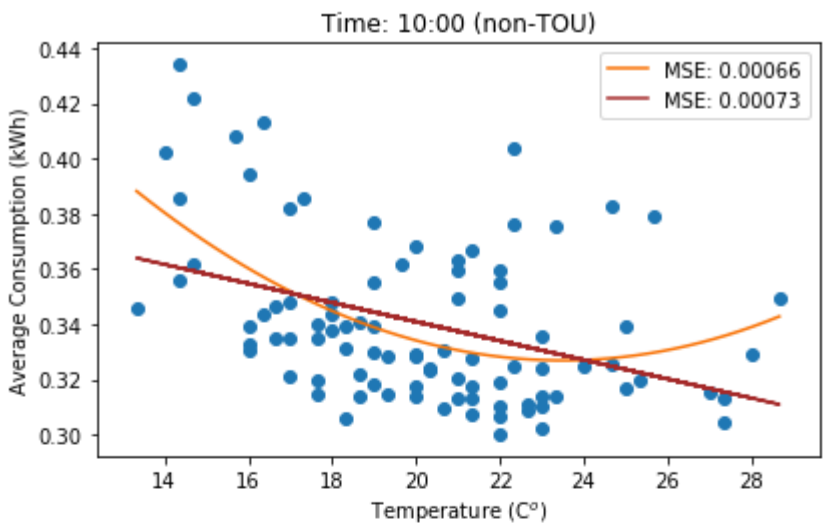

Time: 07:00 (TOU)
MSE: 0.00053
MSE: 0.00053
Average Consumption (kWh)
0.38
0.36
0.34
0.32
0.30
0.28
0.26
10
12
14
16
18
20
22
Temperature (C°)

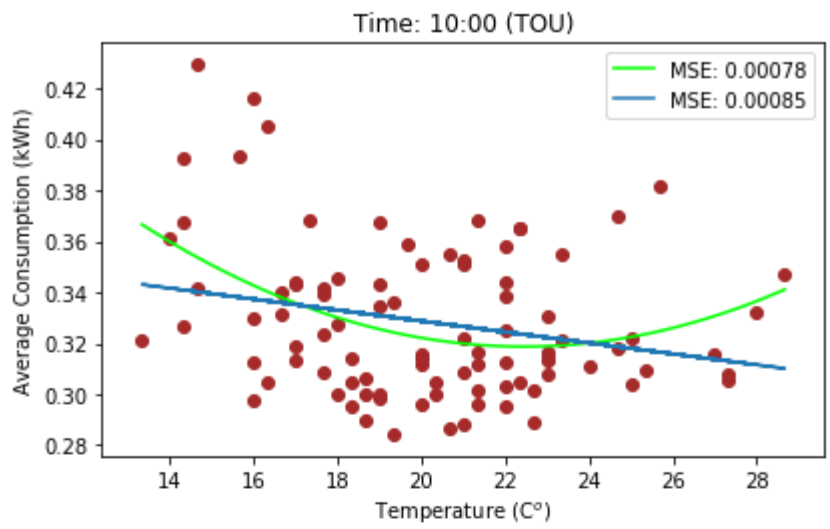

Time: 08:00 (non-TOU)
MSE: 0.00016
MSE: 0.0002
Average Consumption (kWh)
0.38
0.37
0.36
0.35
0.34
0.33
0.32
0.31
12
14
16
18
20
22
24
Temperature (C°)

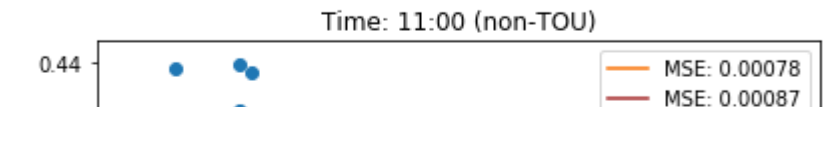

Time: 08:00 (TOU)
MSE: 0.00023
MSE: 0.00026
Average Consumption (kWh)
0.40
0.38
0.36
0.34
0.32
0.30
12
14
16
18
20
22
24
Temperature (C°)

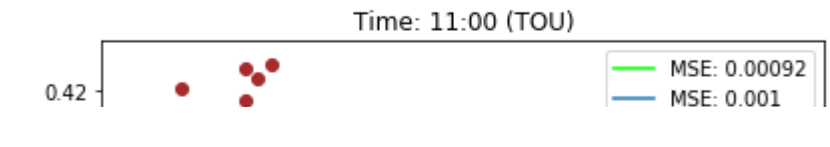

Time: 09:00 (non-TOU)
MSE: 0.00047
MSE: 0.00053
Average Consumption (kWh)
0.42
0.40
0.38
0.36
0.34
0.32
14
16
18
20
22
24
26
Temperature (C°)

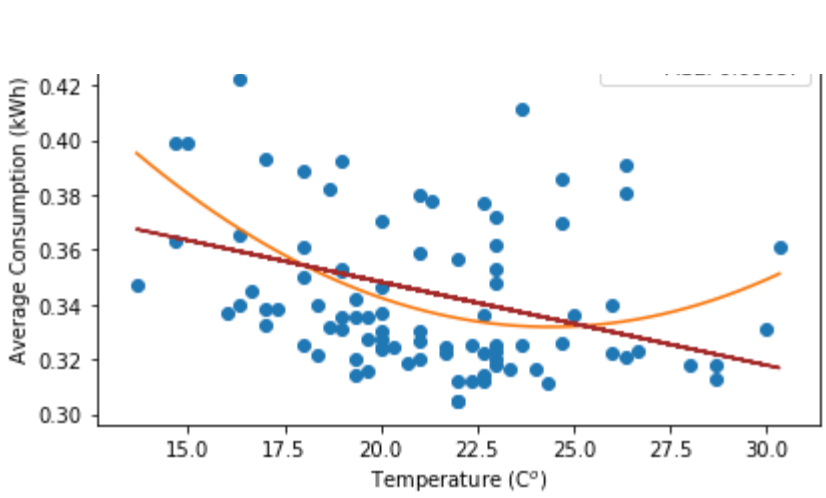

Time: 09:00 (TOU)
MSE: 0.00057
MSE: 0.00064
Average Consumption (kWh)
0.42
0.40
0.38
0.36
0.34
0.32
0.30
14
16
18
20
22
24
26
Temperature (C°)

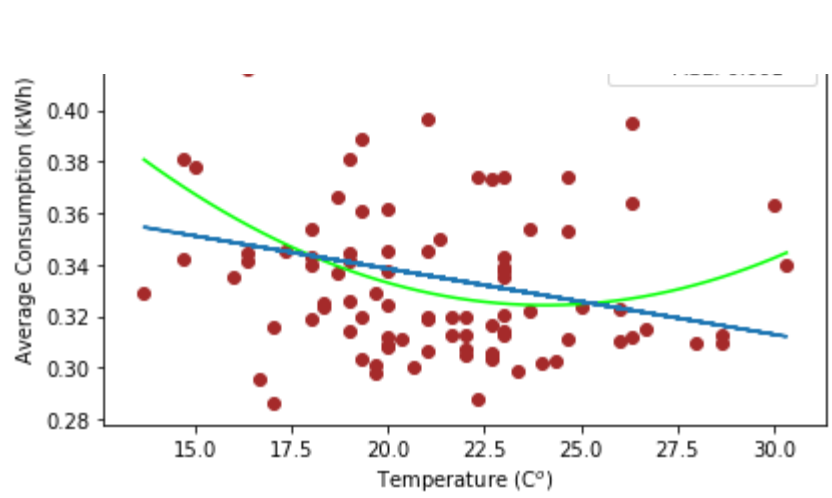

Time: 10:00 (non-TOU)
MSE: 0.00066
MSE: 0.00073
Average Consumption (kWh)
0.44
0.42
0.40
0.38
0.36
0.34
0.32
0.30
14
16
18
20
22
24
26
28
Temperature (C°)

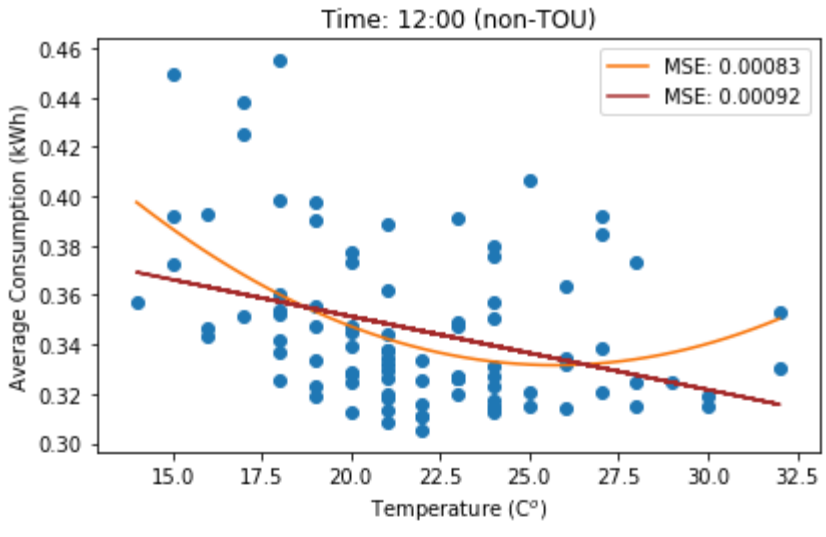

Time: 10:00 (TOU)
MSE: 0.00078
MSE: 0.00085
Average Consumption (kWh)
0.42
0.40
0.38
0.36
0.34
0.32
0.30
0.28
14
16
18
20
22
24
26
28
Temperature (C°)

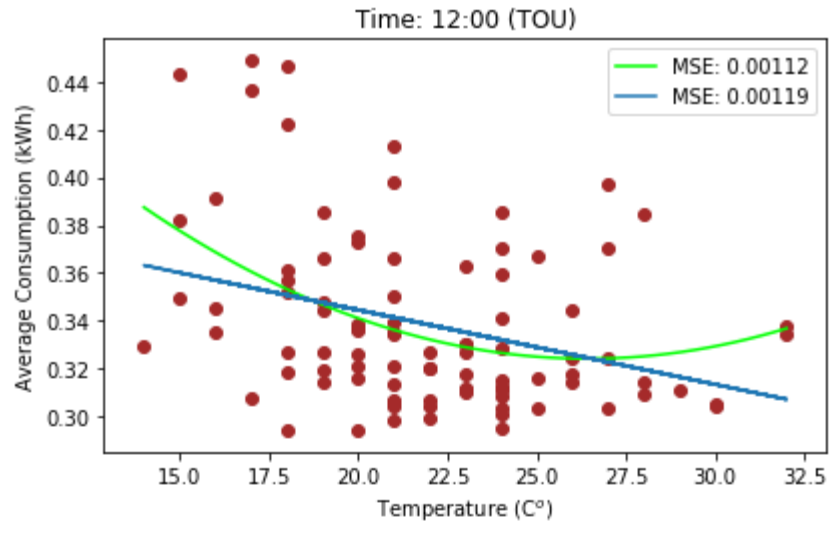

Time: 11:00 (non-TOU)
MSE: 0.00078
0.44

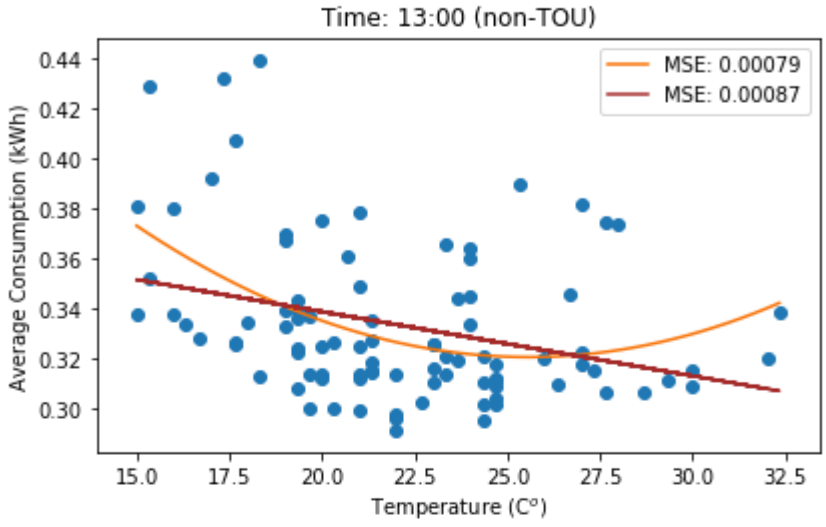

Time: 11:00 (TOU)
MSE: 0.00092
0.42

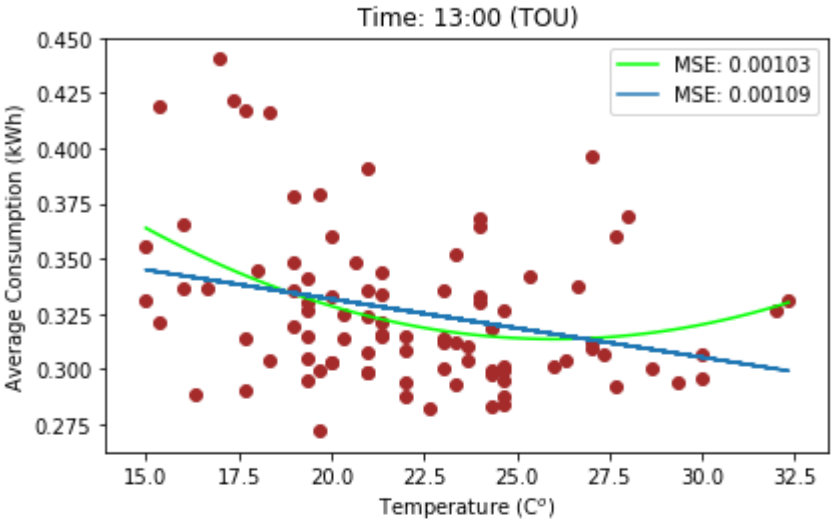

Average Consumption (kWh)
Temperature (C°)

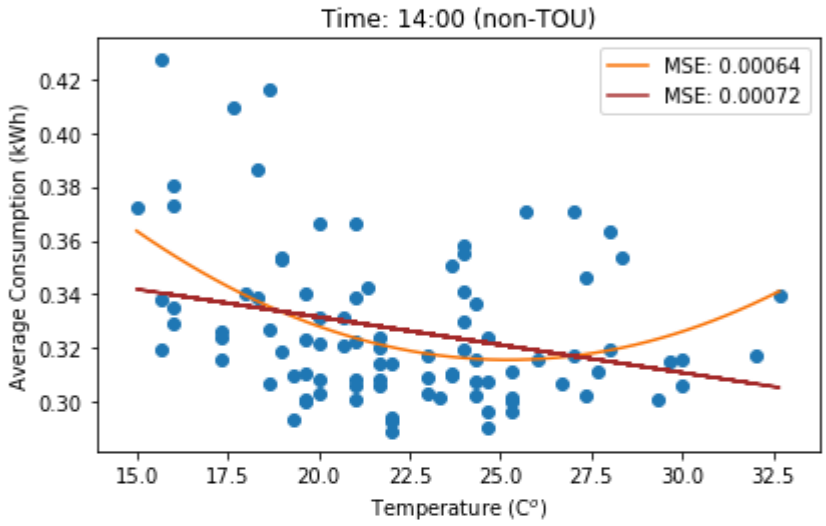

Average Consumption (kWh)
Temperature (C°)

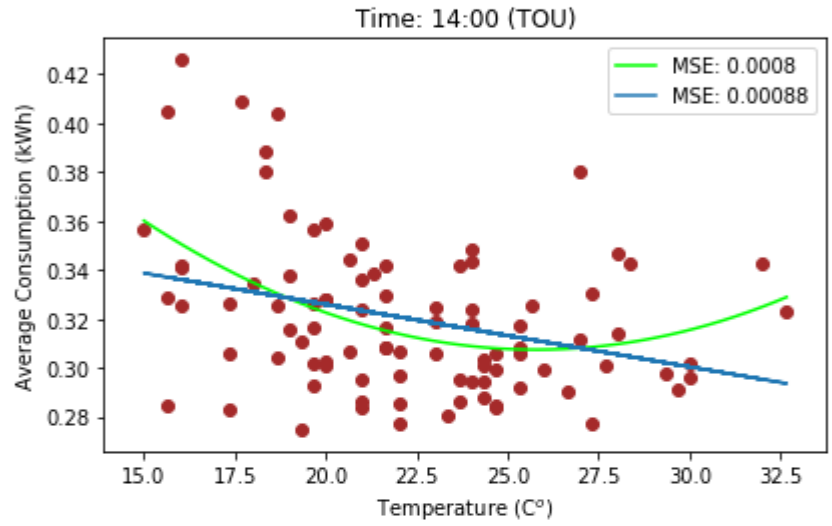

Time: 12:00 (non-TOU)
MSE: 0.00083
MSE: 0.00092
Average Consumption (kWh)
Temperature (C°)

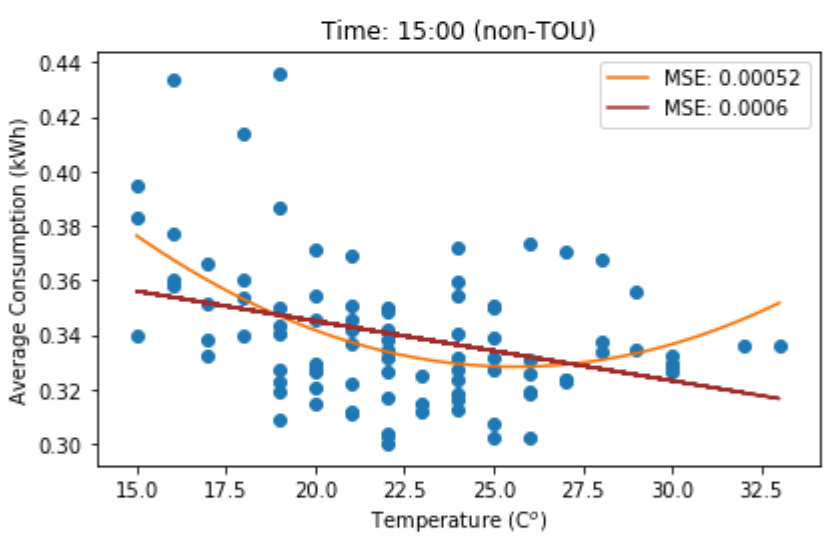

Time: 12:00 (TOU)
MSE: 0.00112
MSE: 0.00119
Average Consumption (kWh)
Temperature (C°)

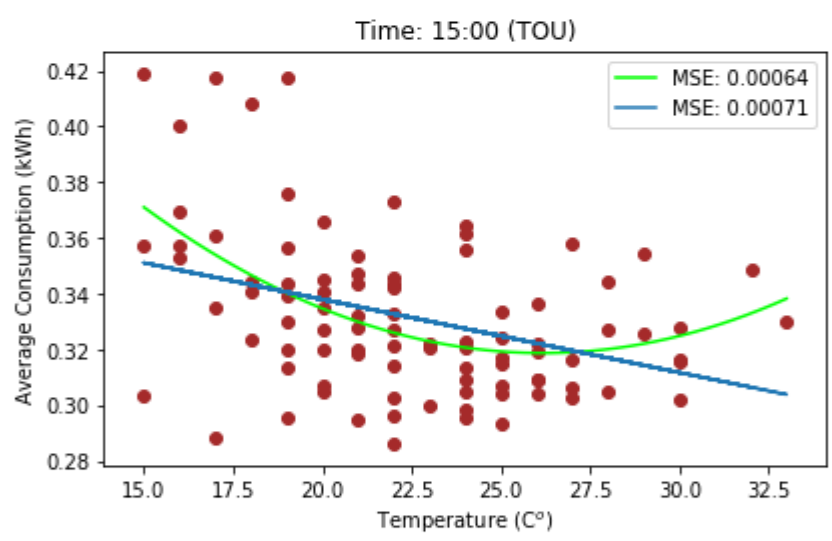

Time: 13:00 (non-TOU)
MSE: 0.00079
MSE: 0.00087
Average Consumption (kWh)
Temperature (C°)

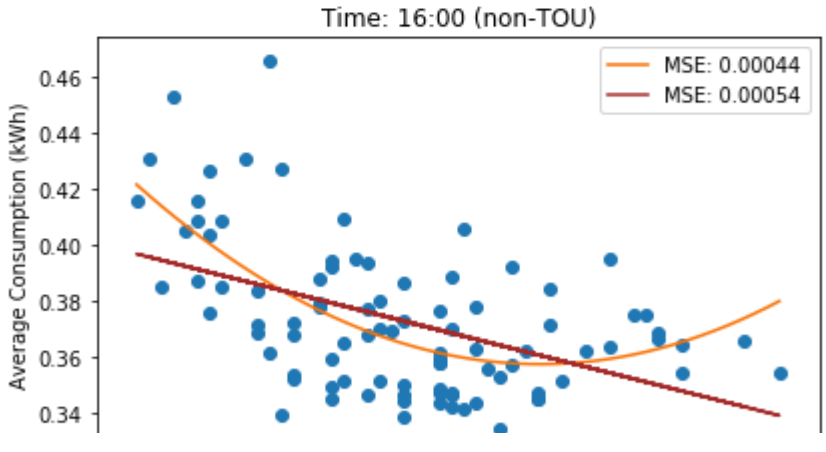

Time: 13:00 (TOU)
MSE: 0.00103
MSE: 0.00109
Average Consumption (kWh)
Temperature (C°)

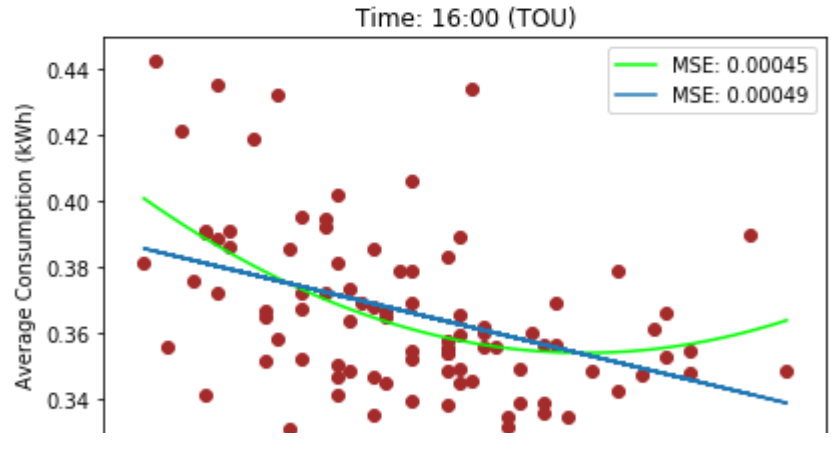

Time: 14:00 (non-TOU)
MSE: 0.00064
MSE: 0.00072
Average Consumption (kWh)
Temperature (C°)

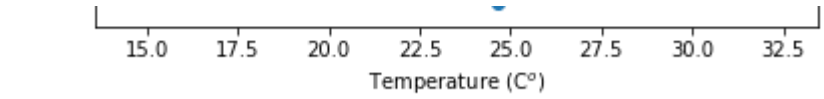

Time: 14:00 (TOU)
MSE: 0.0008
MSE: 0.00088
Average Consumption (kWh)
Temperature (C°)

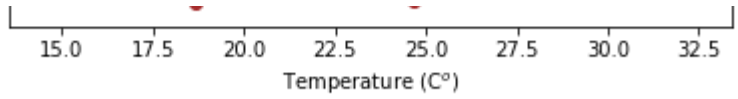

Time: 15:00 (non-TOU)
MSE: 0.00052
MSE: 0.0006
Average Consumption (kWh)
Temperature (C°)

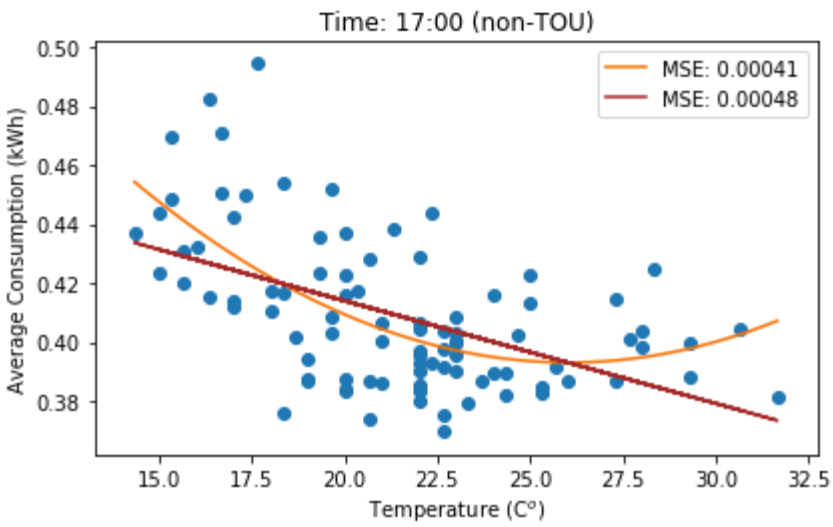

Time: 15:00 (TOU)
MSE: 0.00064
MSE: 0.00071
Average Consumption (kWh)
Temperature (C°)

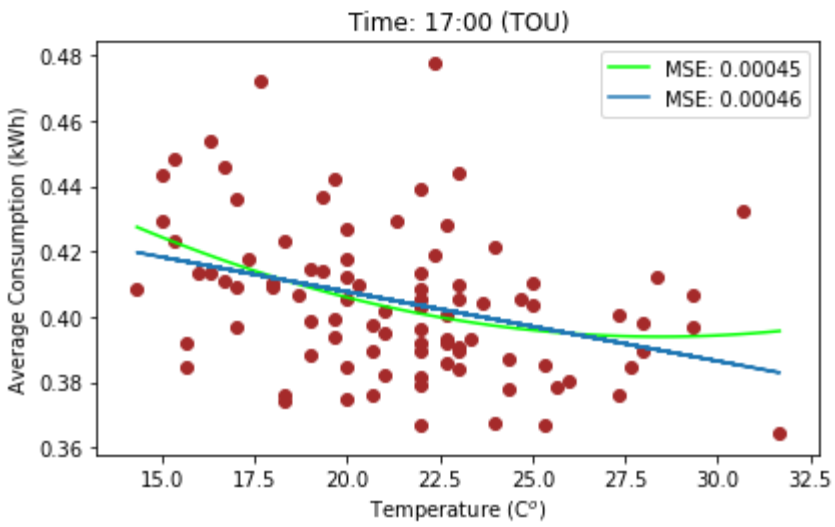

Time: 16:00 (non-TOU)
MSE: 0.00044
MSE: 0.00054
Average Consumption (kWh)

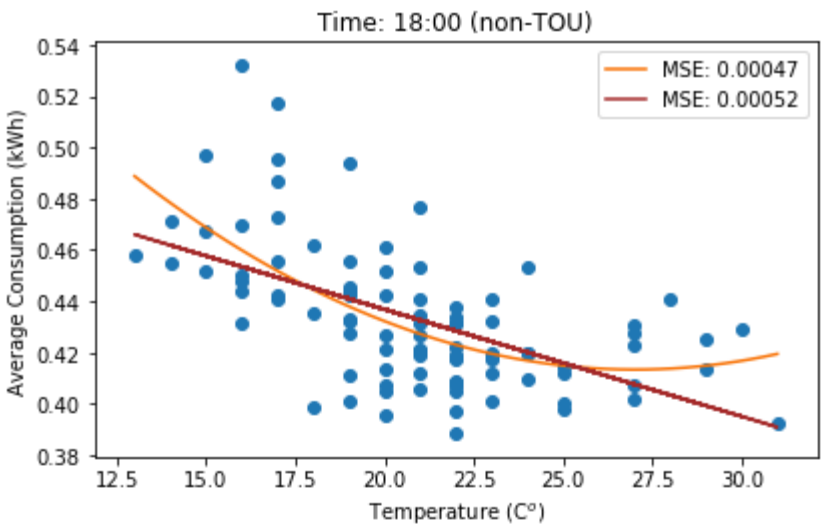

Time: 16:00 (TOU)
MSE: 0.00045
MSE: 0.00049
Average Consumption (kWh)

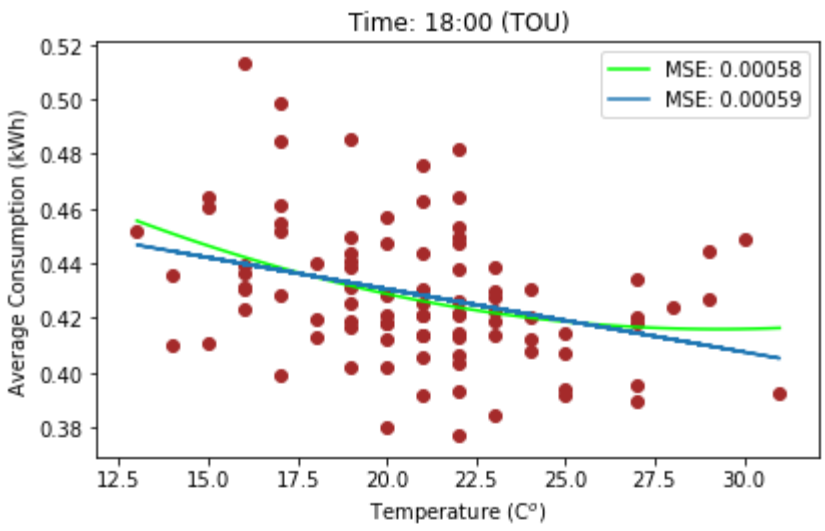

15.0 17.5 20.0 22.5 25.0 27.5 30.0 32.5
Temperature (C°)

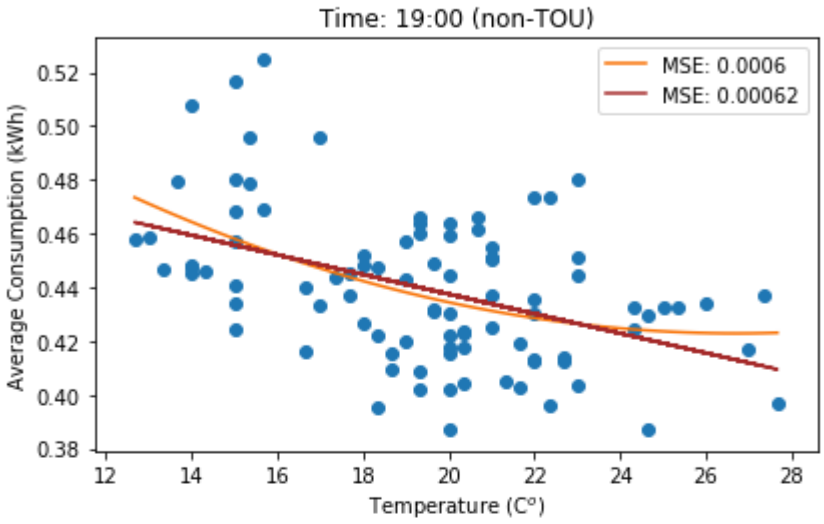

15.0 17.5 20.0 22.5 25.0 27.5 30.0 32.5
Temperature (C°)

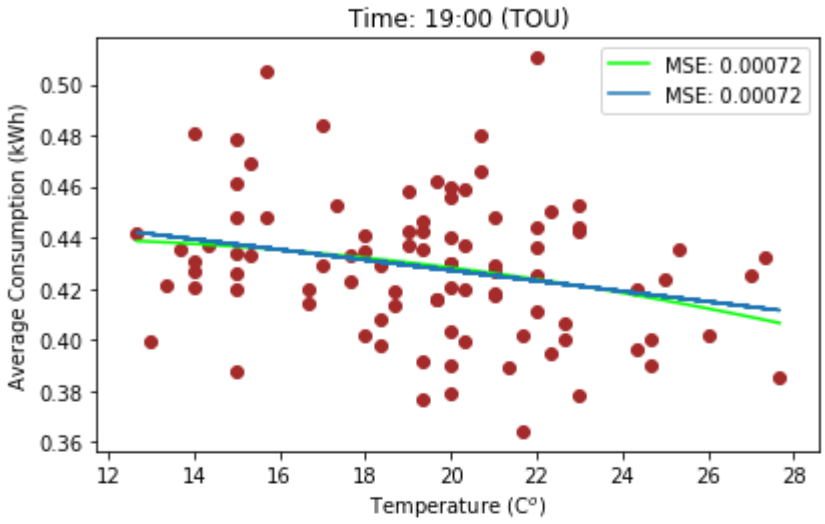

Time: 17:00 (non-TOU)
MSE: 0.00041
MSE: 0.00048
Average Consumption (kWh)
0.50 0.48 0.46 0.44 0.42 0.40 0.38
15.0 17.5 20.0 22.5 25.0 27.5 30.0 32.5
Temperature (C°)

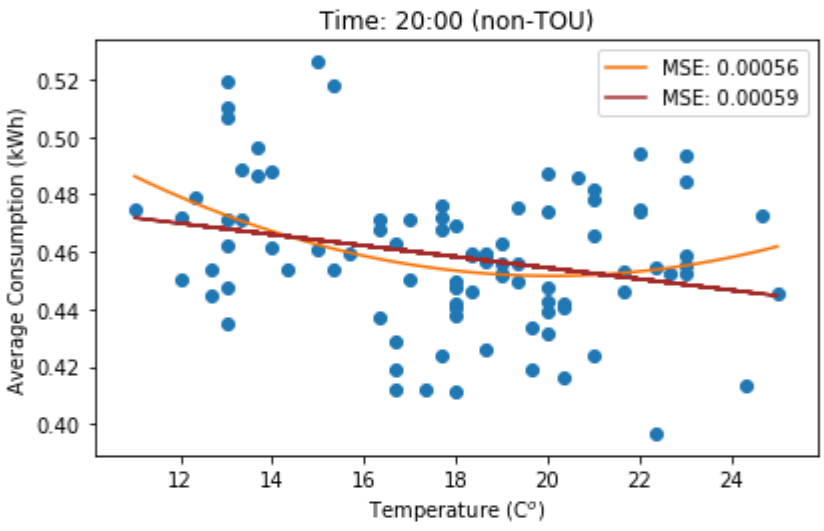

Time: 17:00 (TOU)
MSE: 0.00045
MSE: 0.00046
Average Consumption (kWh)
0.48 0.46 0.44 0.42 0.40 0.38 0.36
15.0 17.5 20.0 22.5 25.0 27.5 30.0 32.5
Temperature (C°)

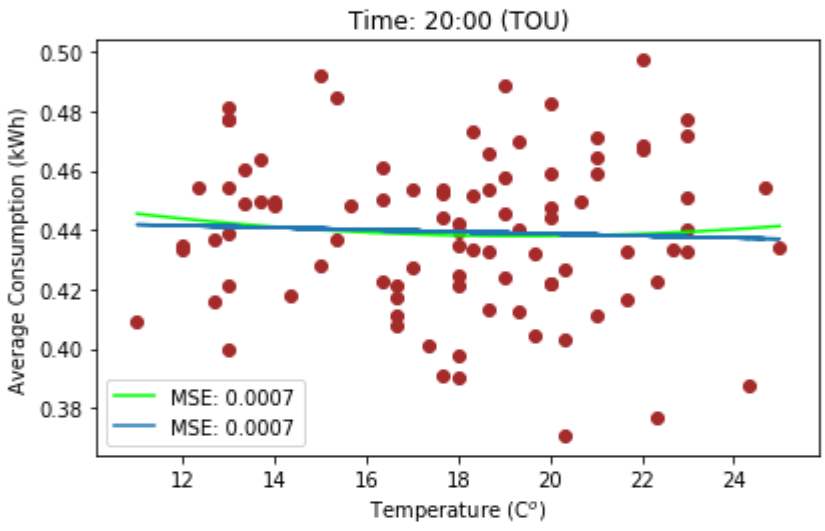

Time: 18:00 (non-TOU)
MSE: 0.00047
MSE: 0.00052
Average Consumption (kWh)
0.54 0.52 0.50 0.48 0.46 0.44 0.42 0.40 0.38
12.5 15.0 17.5 20.0 22.5 25.0 27.5 30.0
Temperature (C°)

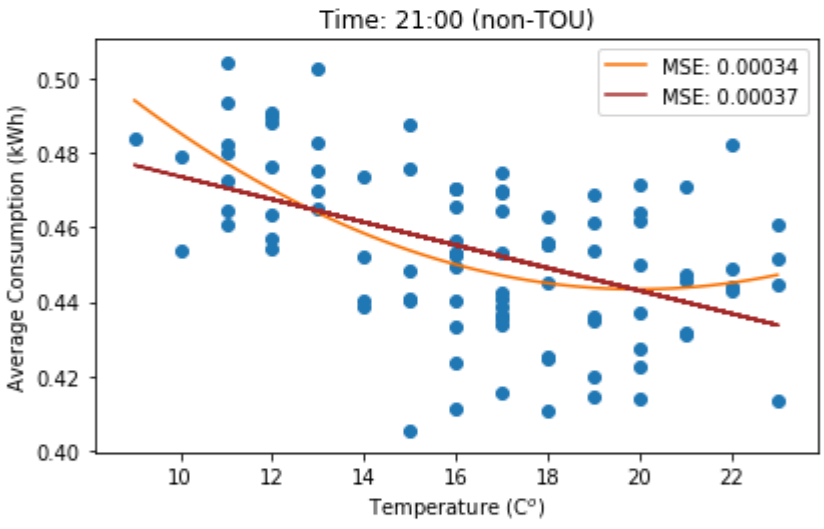

Time: 18:00 (TOU)
MSE: 0.00058
MSE: 0.00059
Average Consumption (kWh)
0.52 0.50 0.48 0.46 0.44 0.42 0.40 0.38
12.5 15.0 17.5 20.0 22.5 25.0 27.5 30.0
Temperature (C°)

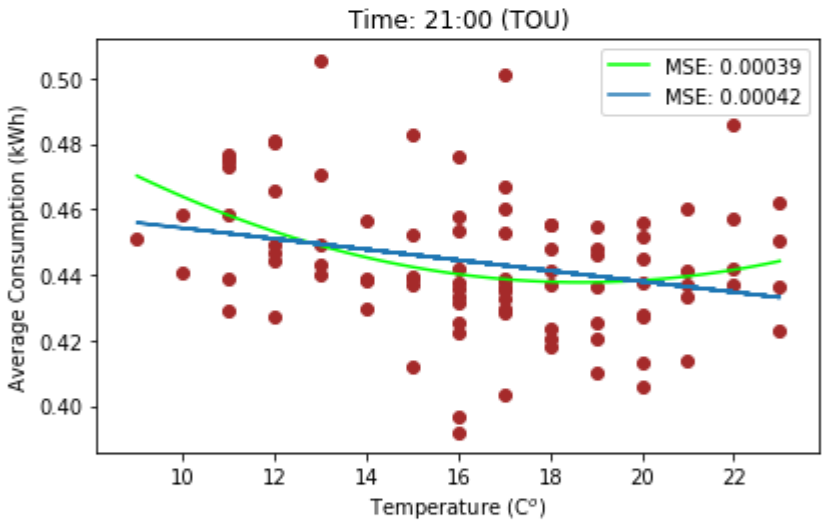

Time: 19:00 (non-TOU)
MSE: 0.0006
MSE: 0.00062
Average Consumption (kWh)
0.52 0.50 0.48 0.46 0.44 0.42 0.40 0.38
12 14 16 18 20 22 24 26 28
Temperature (C°)

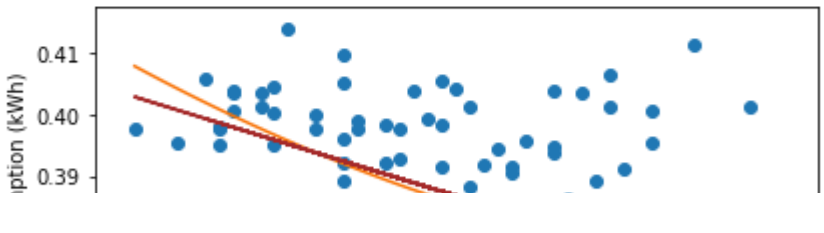

Time: 19:00 (TOU)
MSE: 0.00072
MSE: 0.00072
Average Consumption (kWh)
0.50 0.48 0.46 0.44 0.42 0.40 0.38 0.36
12 14 16 18 20 22 24 26 28
Temperature (C°)

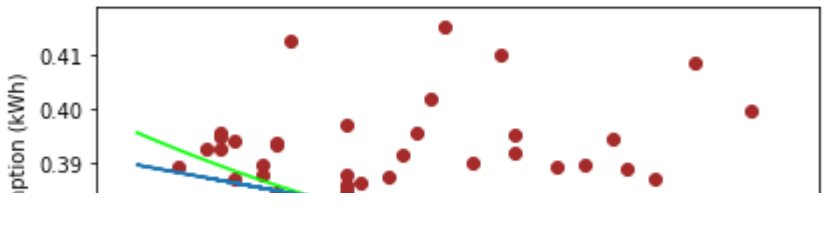

Time: 20:00 (non-TOU)
MSE: 0.00056
MSE: 0.00059
Average Consumption (kWh)
0.52 0.50 0.48 0.46 0.44 0.42 0.40
12 14 16 18 20 22 24
Temperature (C°)

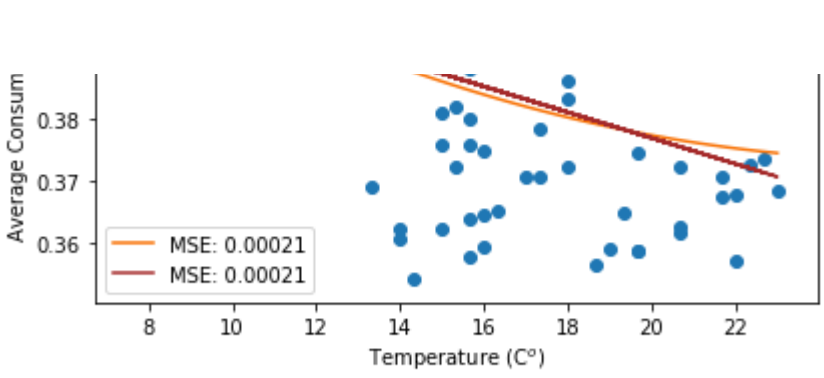

Time: 20:00 (TOU)
MSE: 0.0007
MSE: 0.0007
Average Consumption (kWh)
0.50 0.48 0.46 0.44 0.42 0.40 0.38
12 14 16 18 20 22 24
Temperature (C°)

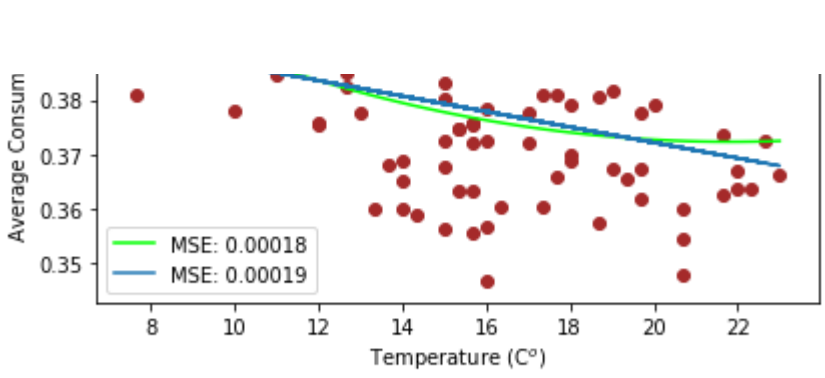

Time: 21:00 (non-TOU)
MSE: 0.00034
MSE: 0.00037
Average Consumption (kWh)
0.50 0.48 0.46 0.44 0.42 0.40
10 12 14 16 18 20 22
Temperature (C°)

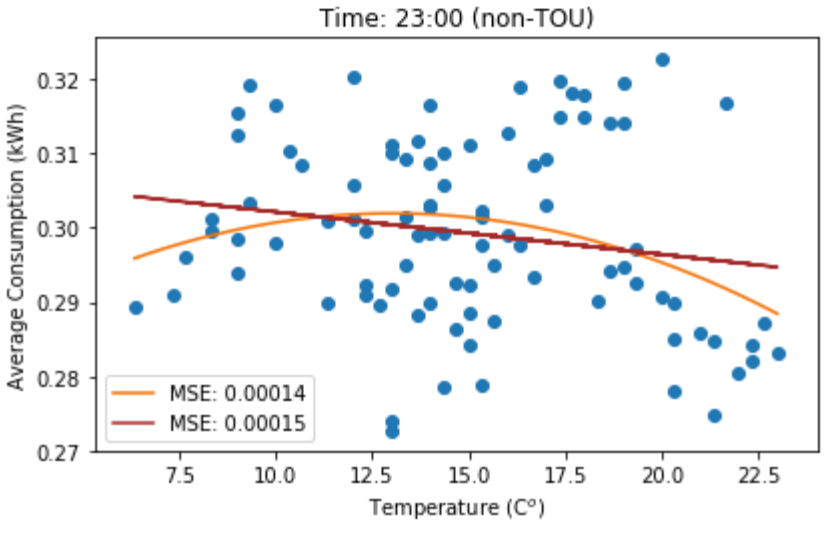

Time: 21:00 (TOU)
MSE: 0.00039
MSE: 0.00042
Average Consumption (kWh)
0.50 0.48 0.46 0.44 0.42 0.40
10 12 14 16 18 20 22
Temperature (C°)

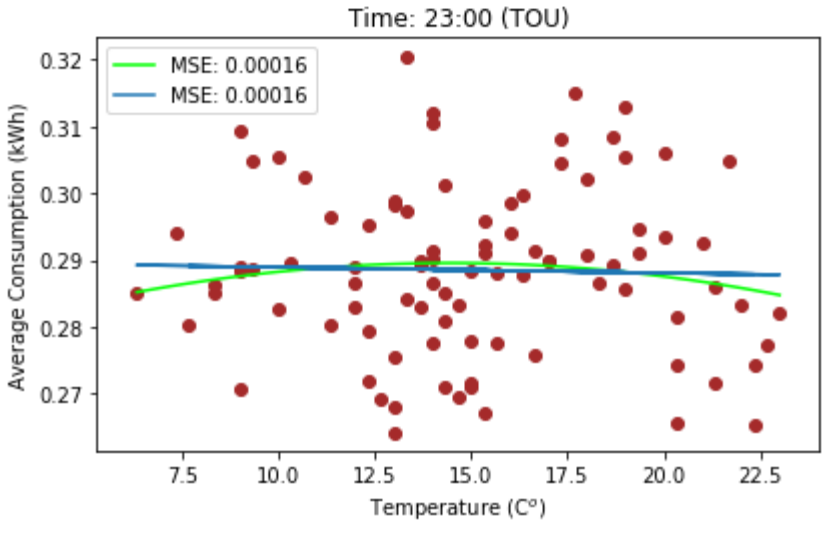

Time: 22:00 (non-TOU)
0.41 0.40 0.39


Time: 22:00 (TOU)
0.41 0.40 0.39

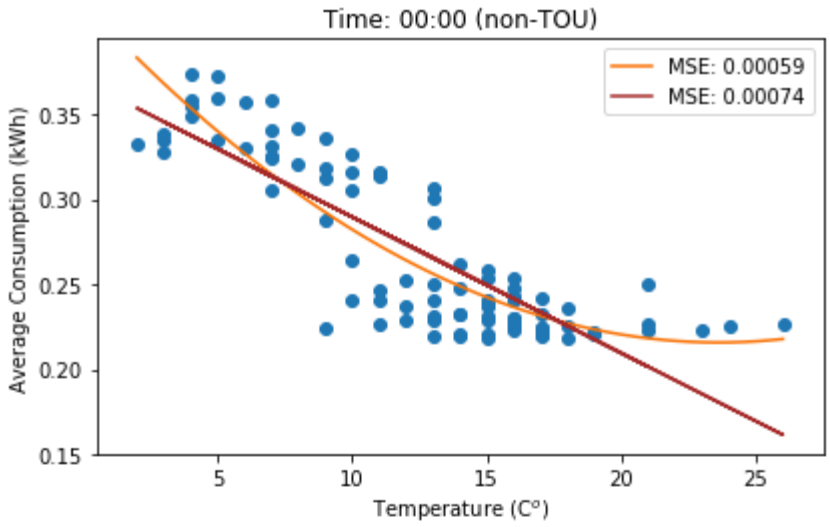

Average Consum
0.38
0.37
0.36
MSE: 0.00021
MSE: 0.00021
8
10
12
14
16
18
20
22
Temperature (C°)

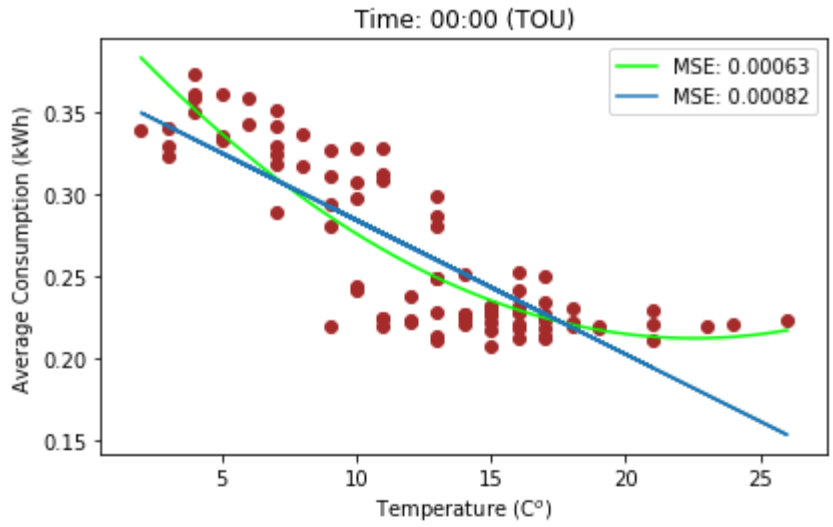

Average Consum
0.38
0.37
0.36
0.35
MSE: 0.00018
MSE: 0.00019
8
10
12
14
16
18
20
22
Temperature (C°)

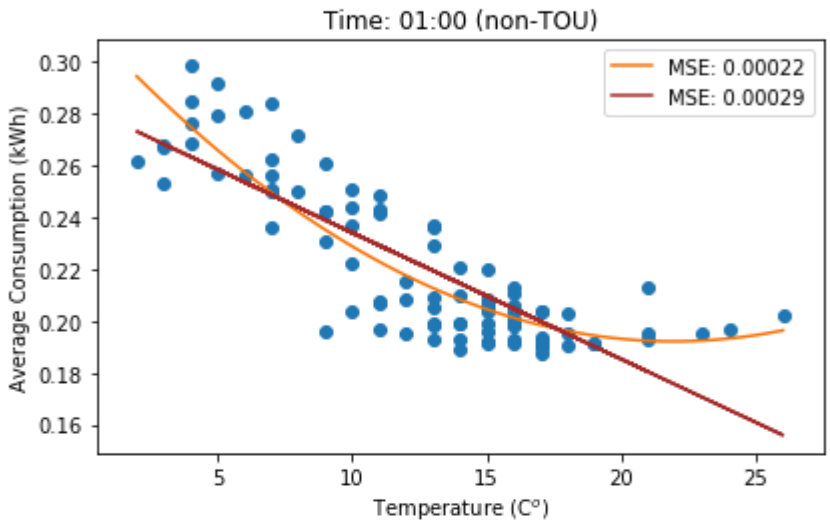

Time: 23:00 (non-TOU)
Average Consumption (kWh)
0.32
0.31
0.30
0.29
0.28
0.27
MSE: 0.00014
MSE: 0.00015
7.5
10.0
12.5
15.0
17.5
20.0
22.5
Temperature (C°)

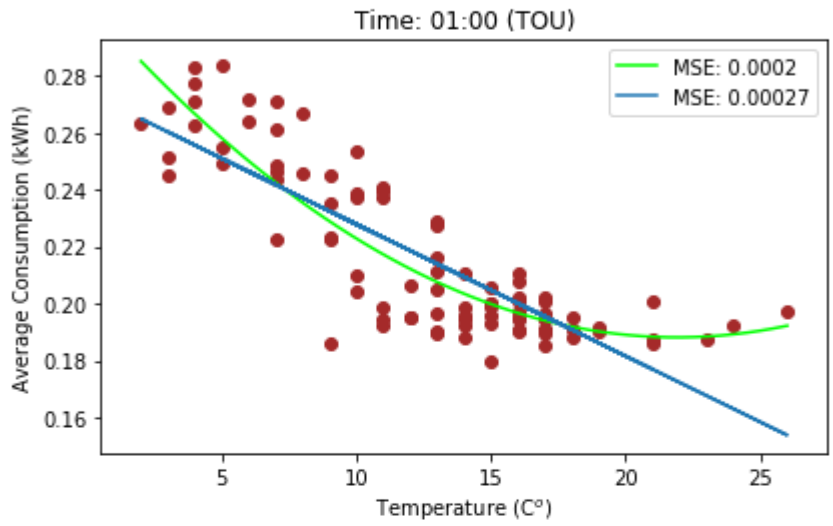

Time: 23:00 (TOU)
Average Consumption (kWh)
0.32
0.31
0.30
0.29
0.28
0.27
MSE: 0.00016
MSE: 0.00016
7.5
10.0
12.5
15.0
17.5
20.0
22.5
Temperature (C°)

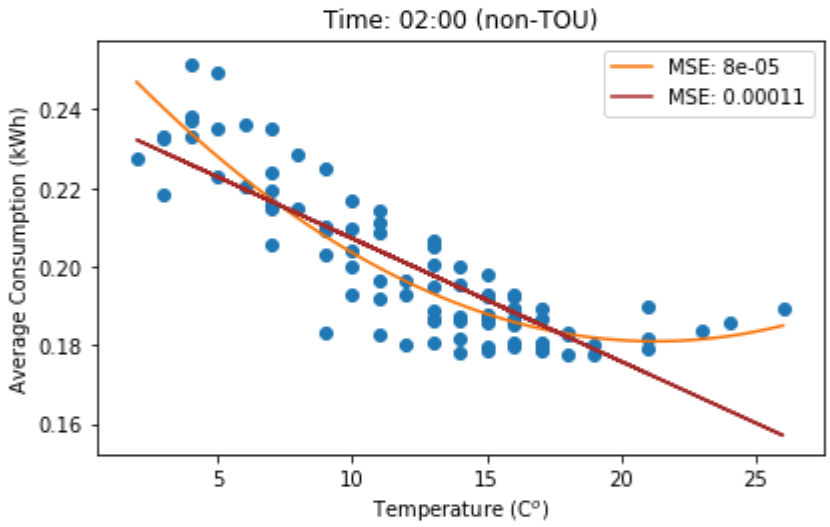
In [21]:
```
# Autumn scatterplot of average consumption (over all consumers at same time period) vs. temp
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
# non-TOU vs TOU in terms of showing the elements in Average column such that they are hourly energy consumpt
ion v.s. temperature over 365 days
fig_all = plt.figure(figsize = (12,90))
ax_Ntou = [] # store subplot objects
ax_tou = [] # store subplot objects
lm = LinearRegression()
for i in range(24):
    ax_Ntou.append(fig_all.add_subplot(24, 2, 2 * i + 1))
    if i <= 9:
        x = df_wea1h_autumn.TempC[df_wea1h_autumn.GMT.str.contains('0' + str(i) + ':00:00')].values
        y = df_Ntou1h_autumn.Average[df_Ntou1h_autumn.GMT.str.contains('0' + str(i) + ':00:00')].values
        xp = np.linspace(min(x), max(x), 50) # polynomial regression
        z = np.polyfit(x, y, 2)
        p = np.poly1d(z)
        ax_Ntou[-1].plot(xp, p(xp), color = 'xkcd:orange', label = 'MSE: ' + str(round(mean_squared_error(y,
p(x)), 5)))
        x = x.reshape(-1, 1)
        y = y.reshape(-1, 1)
        lm.fit(x, y)
        ax_Ntou[-1].plot(x, lm.predict(x), color = 'brown', label = 'MSE: ' + str(round(mean_squared_error(y,
lm.predict(x)), 5)))
        ax_Ntou[-1].scatter(x, y)
        ax_Ntou[-1].set_title('Time: 0' + str(i) + ':00 (non-TOU)')
        ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_Ntou[-1].set_ylabel('Average Consumption (kWh)')
        ax_Ntou[-1].legend()
    else:
        x = df_wea1h_autumn.TempC[df_wea1h_autumn.GMT.str.contains(str(i) + ':00:00')].values
        y = df_Ntou1h_autumn.Average[df_Ntou1h_autumn.GMT.str.contains(str(i) + ':00:00')].values
        xp = np.linspace(min(x), max(x), 50) # polynomial regression
        z = np.polyfit(x, y, 2)
        p = np.poly1d(z)
        ax_Ntou[-1].plot(xp, p(xp), color = 'xkcd:orange', label = 'MSE: ' + str(round(mean_squared_error(y,
p(x)), 5)))
        x = x.reshape(-1, 1)
        y = y.reshape(-1, 1)
        lm.fit(x, y)
        ax_Ntou[-1].plot(x, lm.predict(x), color = 'brown', label = 'MSE: ' + str(round(mean_squared_error(y,
lm.predict(x)), 5)))
        ax_Ntou[-1].scatter(x, y)
        ax_Ntou[-1].set_title('Time: ' + str(i) + ':00 (non-TOU)')
        ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_Ntou[-1].set_ylabel('Average Consumption (kWh)')
        ax_Ntou[-1].legend()
for i in range(24):
    ax_tou.append(fig_all.add_subplot(24, 2, 2 * i + 2))
    if i <= 9:
        x = df_wea1h_autumn.TempC[df_wea1h_autumn.GMT.str.contains('0' + str(i) + ':00:00')].values
        y = df_tou1h_autumn.Average[df_tou1h_autumn.GMT.str.contains('0' + str(i) + ':00:00')].values
        xp = np.linspace(min(x), max(x), 50) # polynomial regression
        z = np.polyfit(x, y, 2)
        p = np.poly1d(z)
        ax_tou[-1].plot(xp, p(xp), color = 'lime', label = 'MSE: ' + str(round(mean_squared_error(y, p(x)), 5
)))
        x = x.reshape(-1, 1)
        y = y.reshape(-1, 1)
        lm.fit(x, y)
        ax_tou[-1].plot(x, lm.predict(x), label = 'MSE: ' + str(round(mean_squared_error(y, lm.predict(x)), 5
)))
        ax_tou[-1].scatter(x, y, color = 'brown')
        ax_tou[-1].set_title('Time: 0' + str(i) + ':00 (TOU)')
        ax_tou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_tou[-1].set_ylabel('Average Consumption (kWh)')
        ax_tou[-1].legend()
    else:
        x = df_wea1h_autumn.TempC[df_wea1h_autumn.GMT.str.contains(str(i) + ':00:00')].values
        y = df_tou1h_autumn.Average[df_tou1h_autumn.GMT.str.contains(str(i) + ':00:00')].values
        xp = np.linspace(min(x), max(x), 50) # polynomial regression
        z = np.polyfit(x, y, 2)
        p = np.poly1d(z)
        ax_tou[-1].plot(xp, p(xp), color = 'lime', label = 'MSE: ' + str(round(mean_squared_error(y, p(x)), 5
)))
        x = x.reshape(-1, 1)
        y = y.reshape(-1, 1)
        lm.fit(x, y)
        ax_tou[-1].plot(x, lm.predict(x), label = 'MSE: ' + str(round(mean_squared_error(y, lm.predict(x)), 5
)))
        ax_tou[-1].scatter(x, y, color = 'brown')
        ax_tou[-1].set_title('Time: ' + str(i) + ':00 (TOU)')
        ax_tou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_tou[-1].set_ylabel('Average Consumption (kWh)')
        ax_tou[-1].legend()
plt.tight_layout()
```

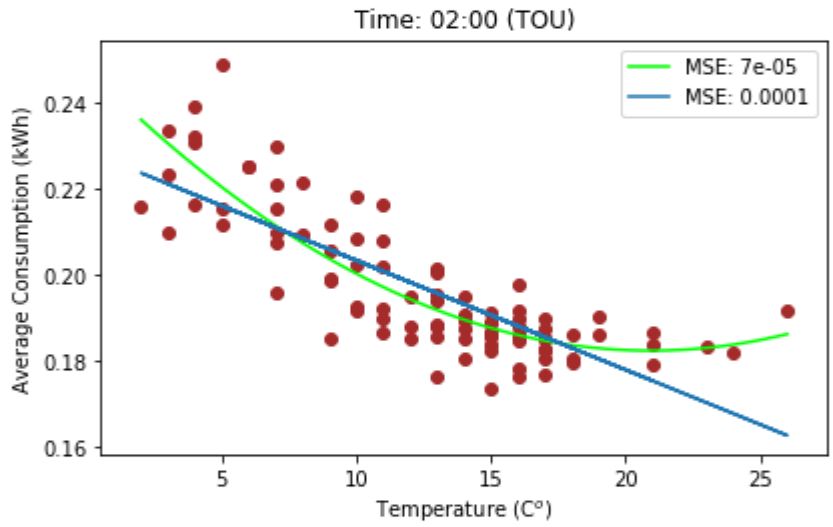

Time: 00:00 (non-TOU)
MSE: 0.00059
MSE: 0.00074
Average Consumption (kWh)
Temperature (C°)

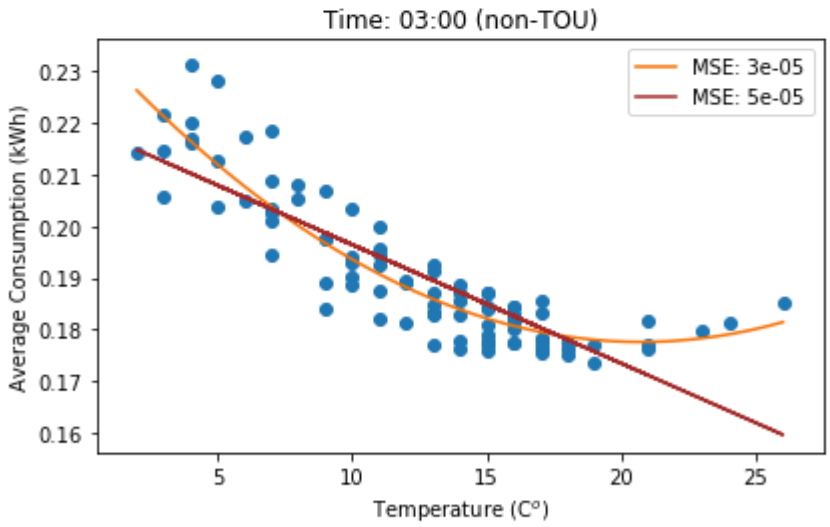

Time: 00:00 (TOU)
MSE: 0.00063
MSE: 0.00082
Average Consumption (kWh)
Temperature (C°)

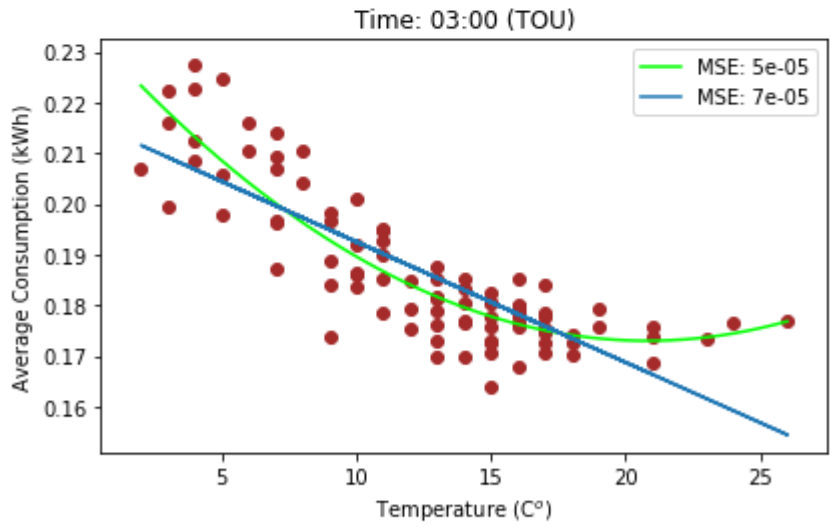

Time: 01:00 (non-TOU)
MSE: 0.00022
MSE: 0.00029
Average Consumption (kWh)
Temperature (C°)

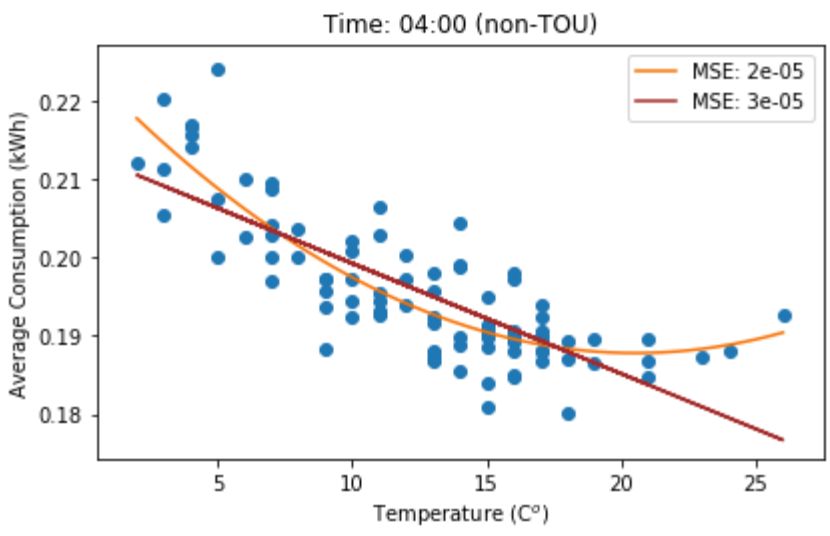

Time: 01:00 (TOU)
MSE: 0.0002
MSE: 0.00027
Average Consumption (kWh)
Temperature (C°)

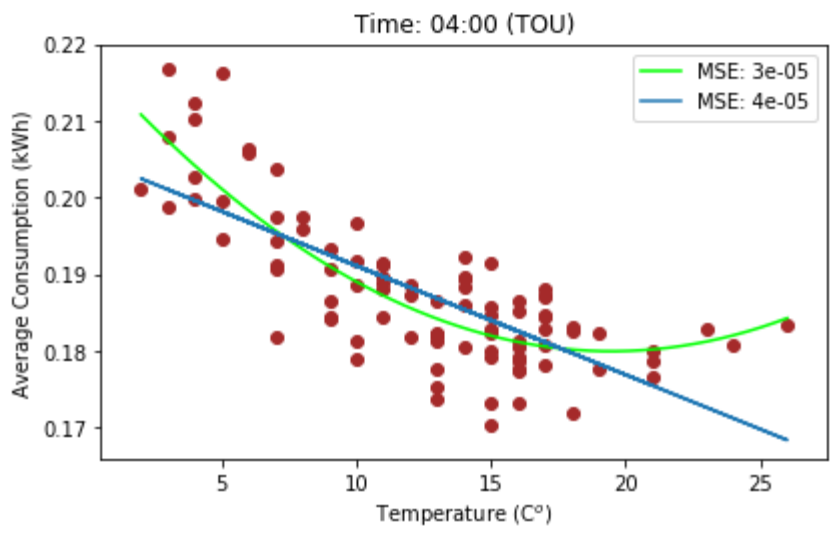

Time: 02:00 (non-TOU)
MSE: 8e-05
MSE: 0.00011
Average Consumption (kWh)
Temperature (C°)

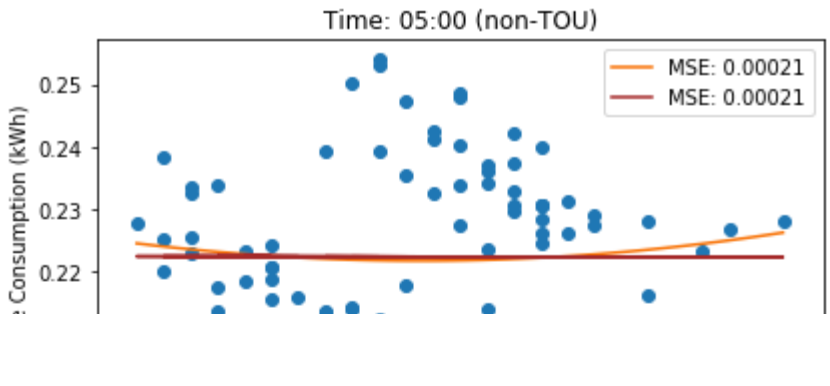

Time: 02:00 (TOU)
MSE: 7e-05
MSE: 0.0001
Average Consumption (kWh)
Temperature (C°)

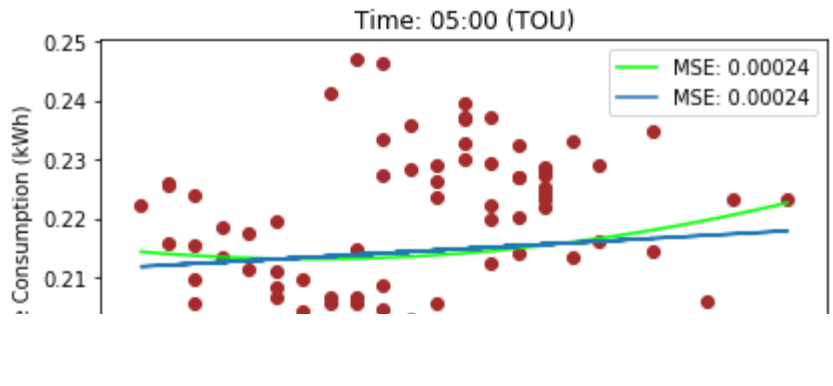

Time: 03:00 (non-TOU)
MSE: 3e-05
MSE: 5e-05
Average Consumption (kWh)
Temperature (C°)

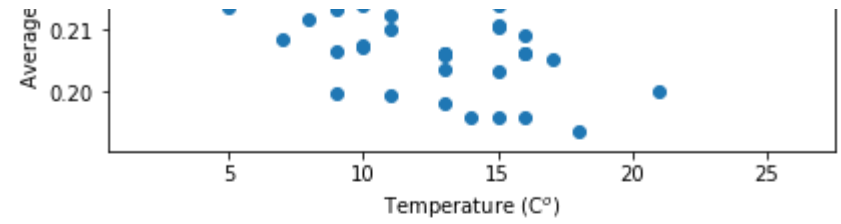

Time: 03:00 (TOU)
MSE: 5e-05
MSE: 7e-05
Average Consumption (kWh)
Temperature (C°)

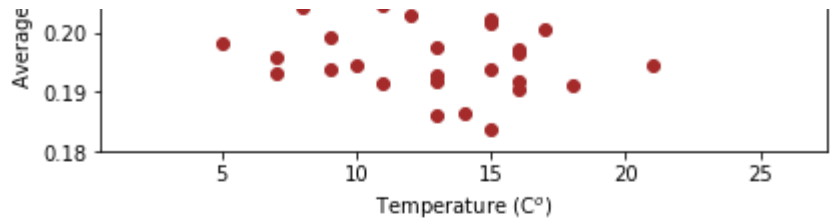

Time: 04:00 (non-TOU)
MSE: 2e-05
MSE: 3e-05
Average Consumption (kWh)
Temperature (C°)

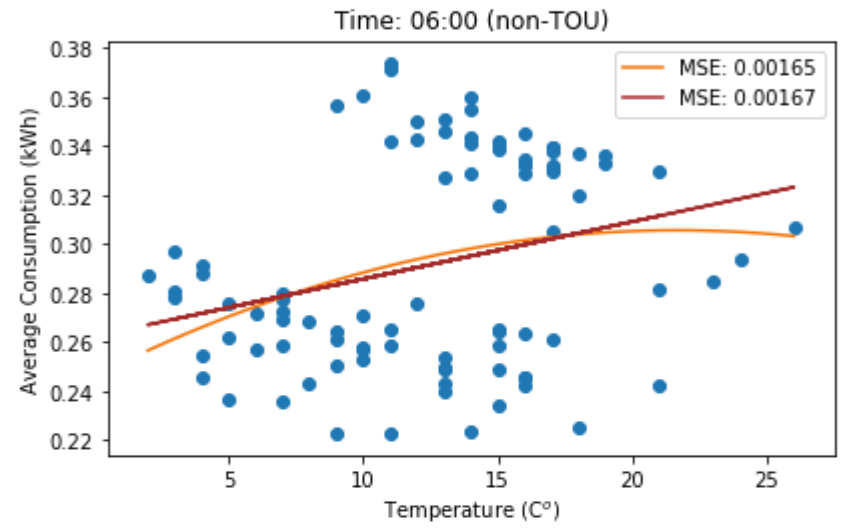

Time: 04:00 (TOU)
MSE: 3e-05
MSE: 4e-05
Average Consumption (kWh)
Temperature (C°)

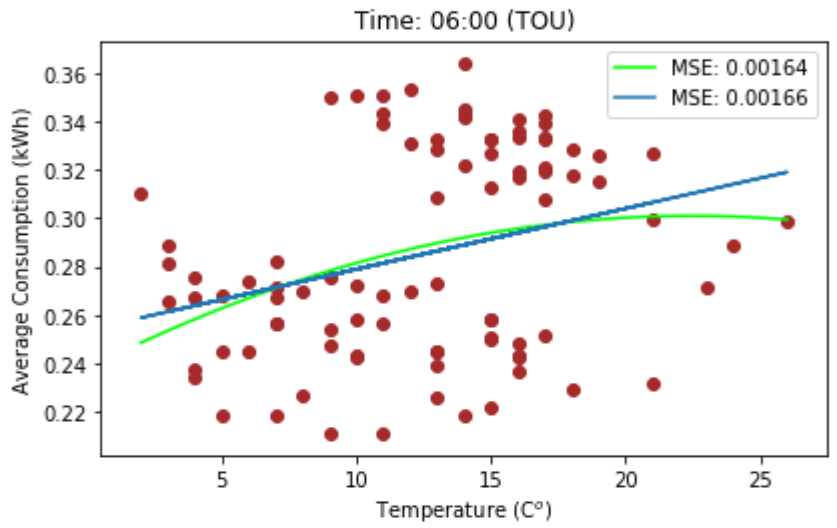

Time: 05:00 (non-TOU)
MSE: 0.00021
MSE: 0.00021
Temperature (C°)

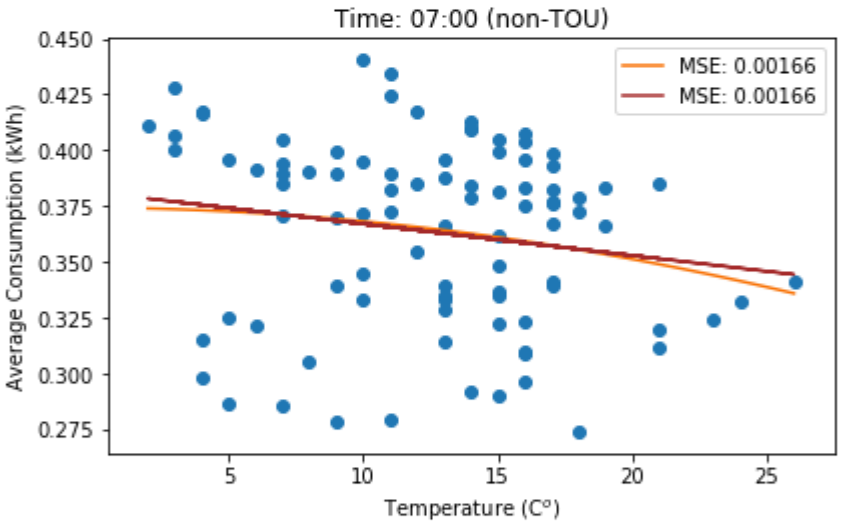

Time: 05:00 (TOU)
MSE: 0.00024
MSE: 0.00024
Temperature (C°)

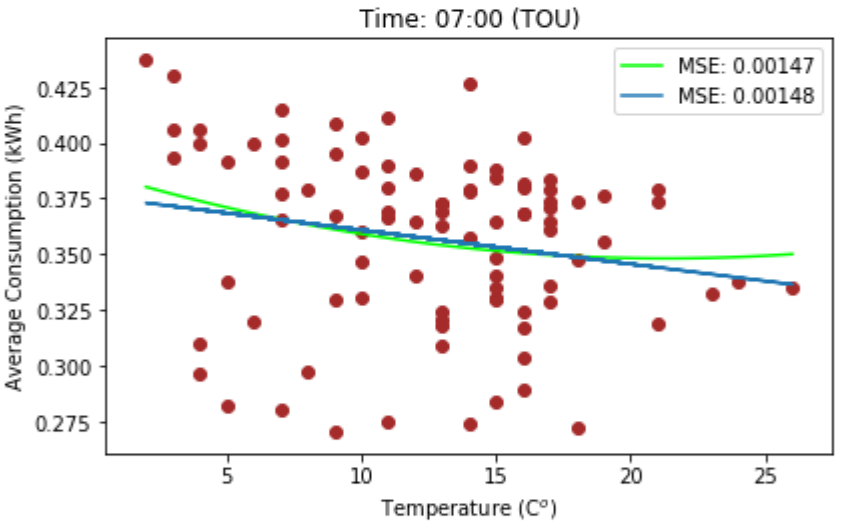

Average
0.21
0.20
5
10
15
20
25
Temperature (C°)

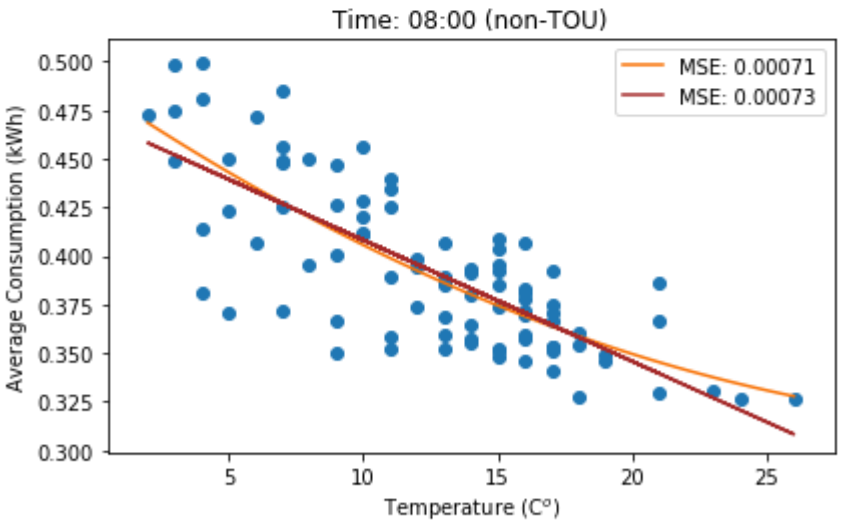

Average
0.20
0.19
0.18
5
10
15
20
25
Temperature (C°)

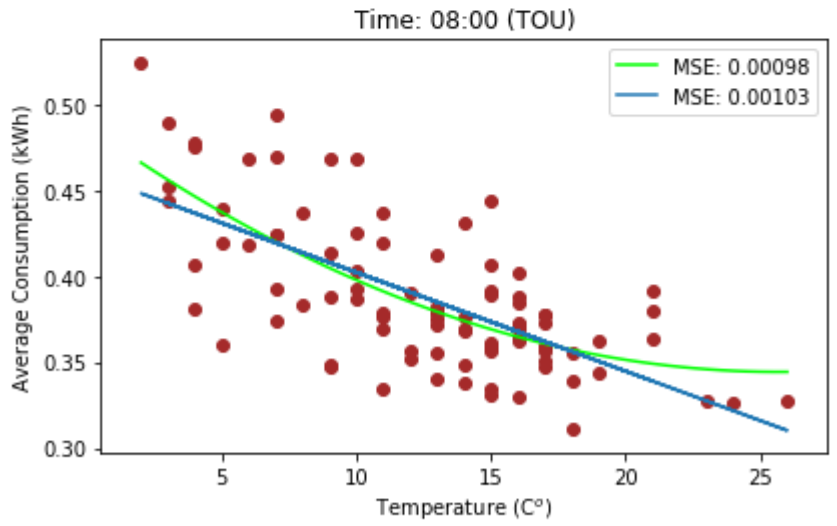

Time: 06:00 (non-TOU)
Average Consumption (kWh)
0.38
0.36
0.34
0.32
0.30
0.28
0.26
0.24
0.22
MSE: 0.00165
MSE: 0.00167
5
10
15
20
25
Temperature (C°)

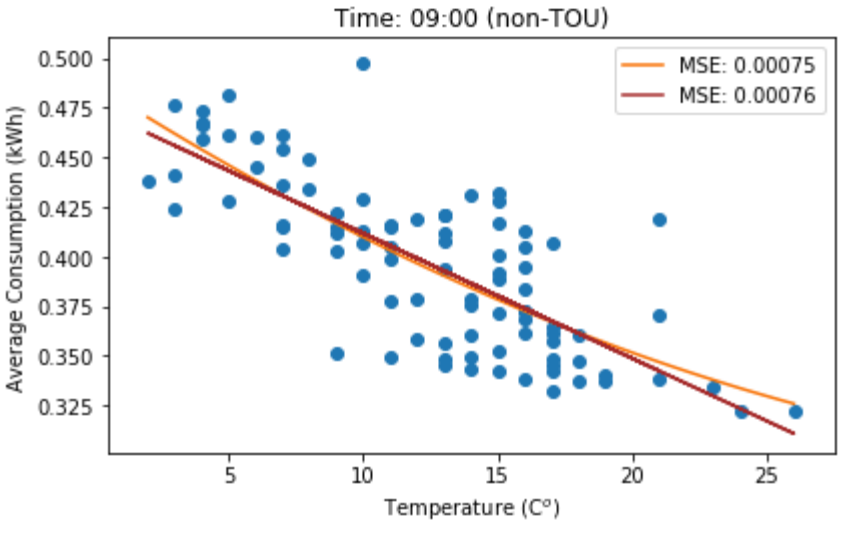

Time: 06:00 (TOU)
Average Consumption (kWh)
0.36
0.34
0.32
0.30
0.28
0.26
0.24
0.22
MSE: 0.00164
MSE: 0.00166
5
10
15
20
25
Temperature (C°)

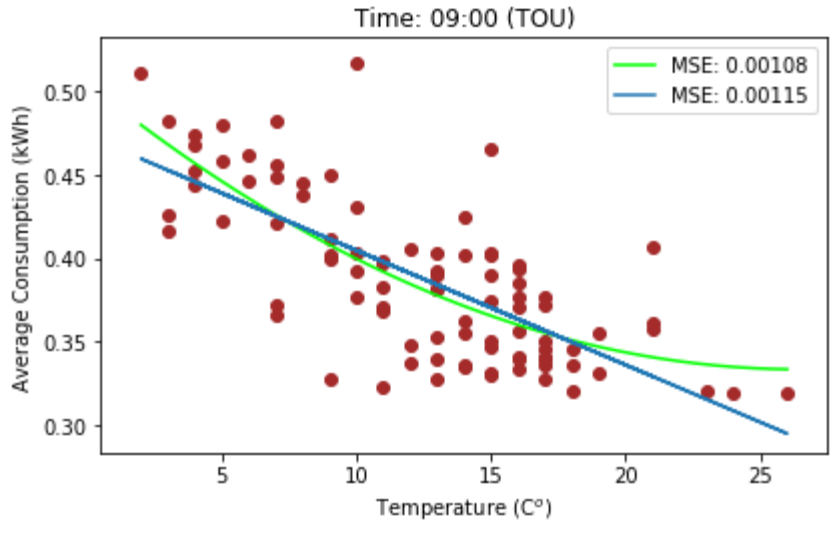

Time: 07:00 (non-TOU)
Average Consumption (kWh)
0.450
0.425
0.400
0.375
0.350
0.325
0.300
0.275
MSE: 0.00166
MSE: 0.00166
5
10
15
20
25
Temperature (C°)

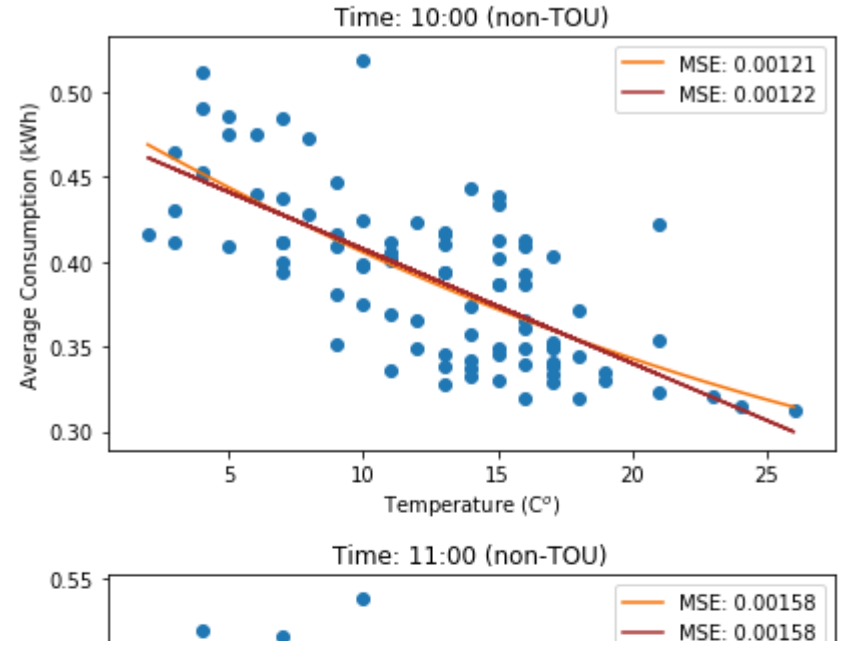

Time: 07:00 (TOU)
Average Consumption (kWh)
0.425
0.400
0.375
0.350
0.325
0.300
0.275
MSE: 0.00147
MSE: 0.00148
5
10
15
20
25
Temperature (C°)

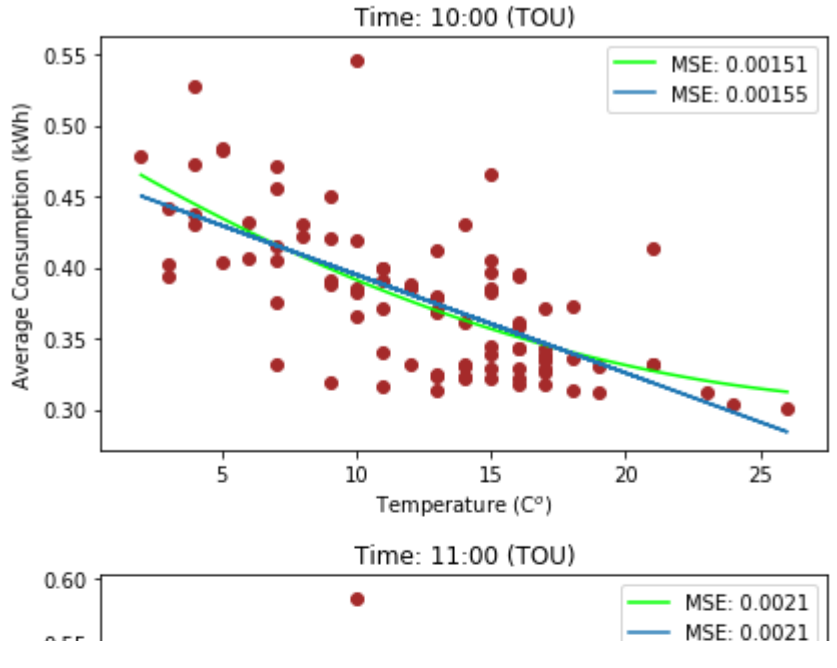

Time: 08:00 (non-TOU)
Average Consumption (kWh)
0.500
0.475
0.450
0.425
0.400
0.375
0.350
0.325
0.300
MSE: 0.00071
MSE: 0.00073
5
10
15
20
25
Temperature (C°)


Time: 08:00 (TOU)
Average Consumption (kWh)
0.50
0.45
0.40
0.35
0.30
MSE: 0.00098
MSE: 0.00103
5
10
15
20
25
Temperature (C°)


Time: 09:00 (non-TOU)
Average Consumption (kWh)
0.500
0.475
0.450
0.425
0.400
0.375
0.350
0.325
MSE: 0.00075
MSE: 0.00076
5
10
15
20
25
Temperature (C°)

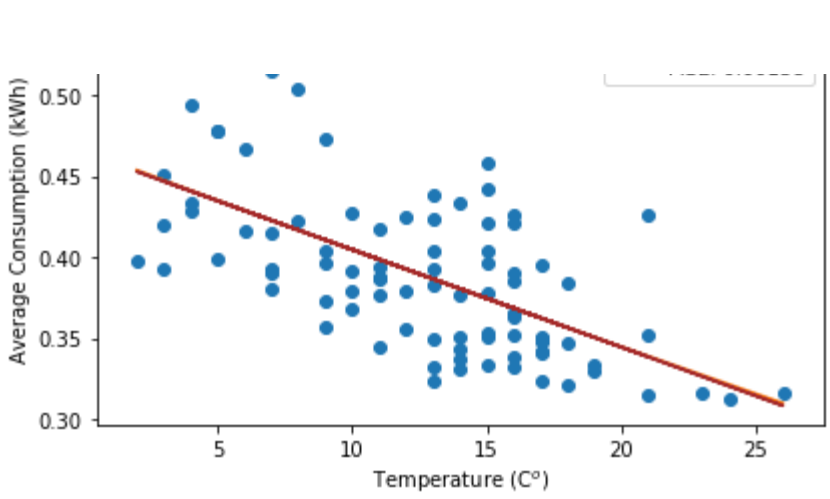

Time: 09:00 (TOU)
Average Consumption (kWh)
0.50
0.45
0.40
0.35
0.30
MSE: 0.00108
MSE: 0.00115
5
10
15
20
25
Temperature (C°)

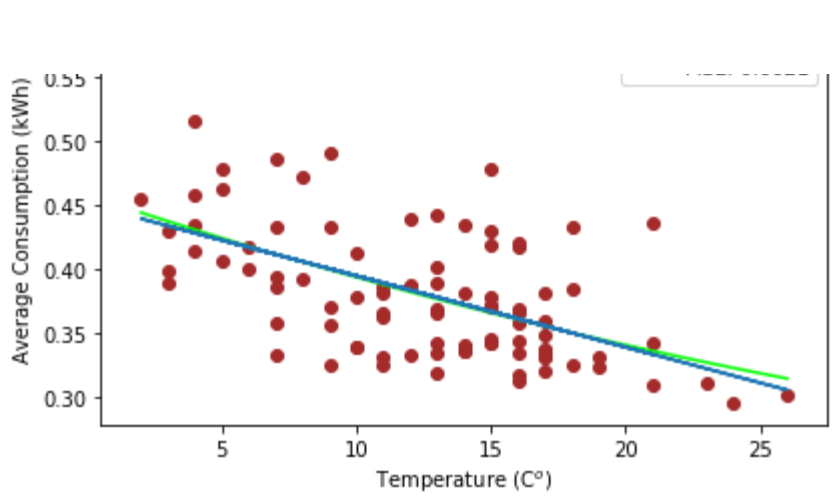

Time: 10:00 (non-TOU)
Average Consumption (kWh)
0.50
0.45
0.40
0.35
0.30
MSE: 0.00121
MSE: 0.00122
5
10
15
20
25
Temperature (C°)

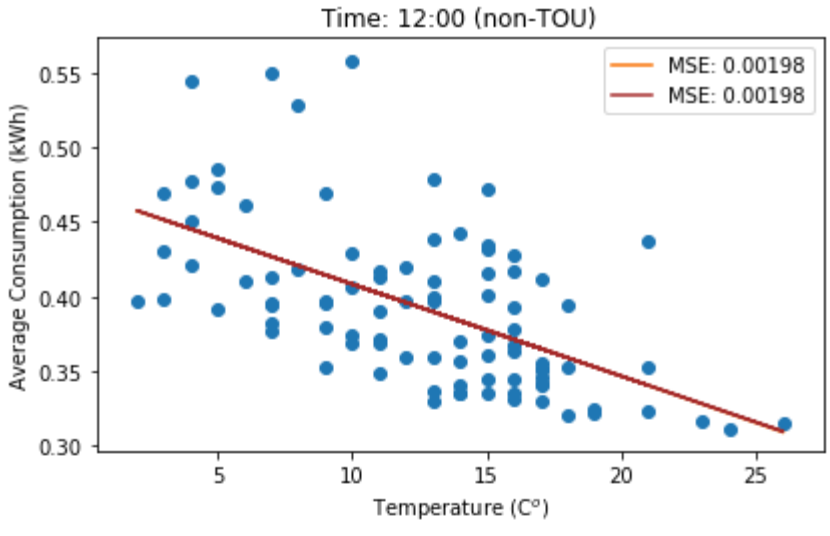

Time: 10:00 (TOU)
Average Consumption (kWh)
0.55
0.50
0.45
0.40
0.35
0.30
MSE: 0.00151
MSE: 0.00155
5
10
15
20
25
Temperature (C°)

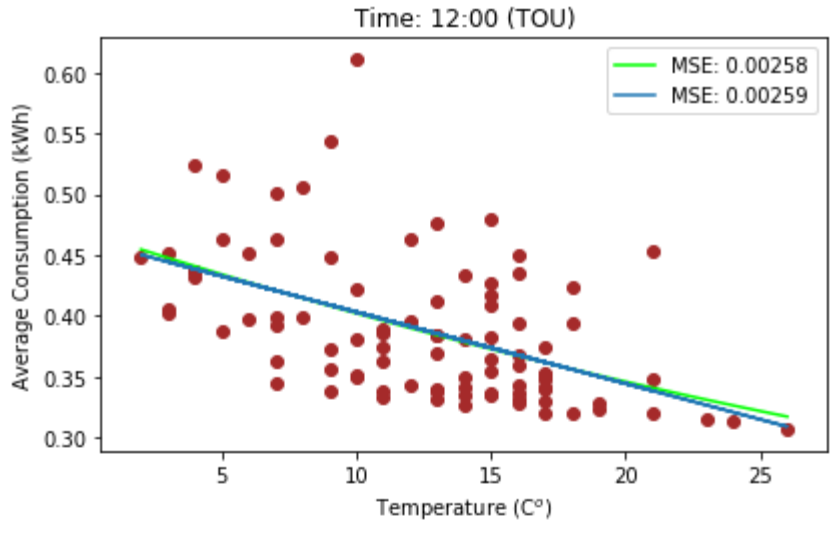
Time: 11:00 (non-TOU)
0.55
MSE: 0.00158
MSE: 0.00158

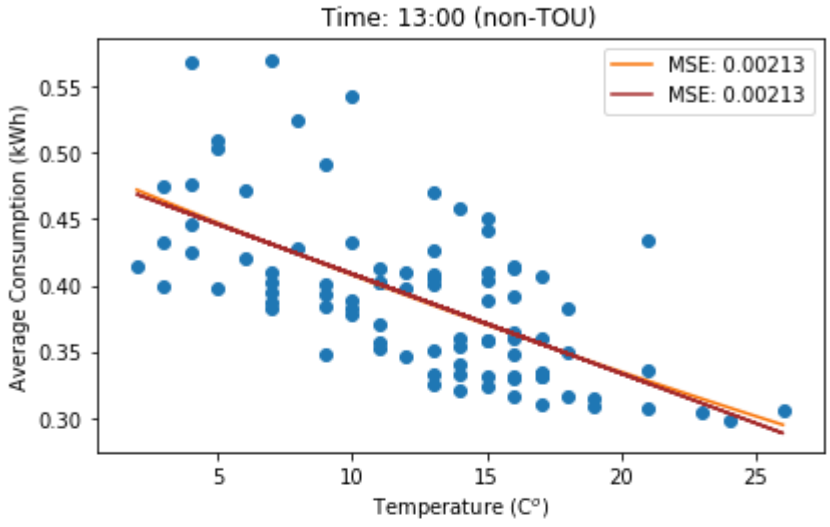
Time: 11:00 (TOU)
0.60
MSE: 0.0021
MSE: 0.0021

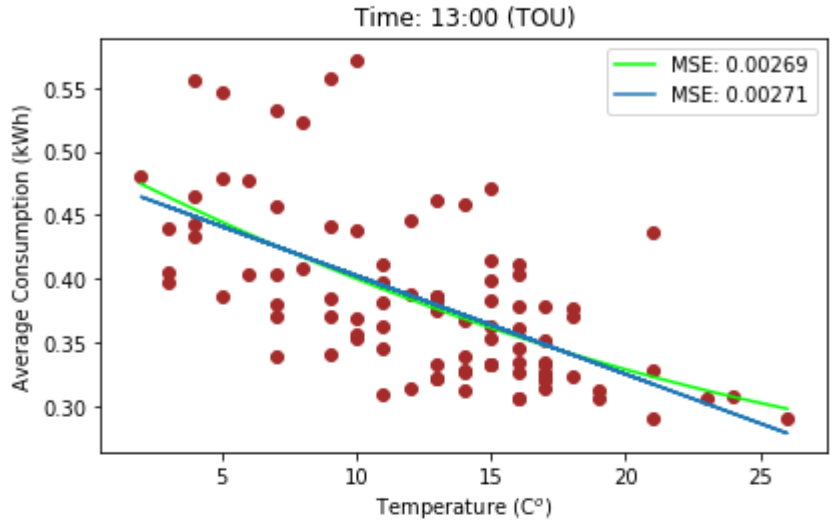

Average Consumption (kWh)
Temperature (C°)

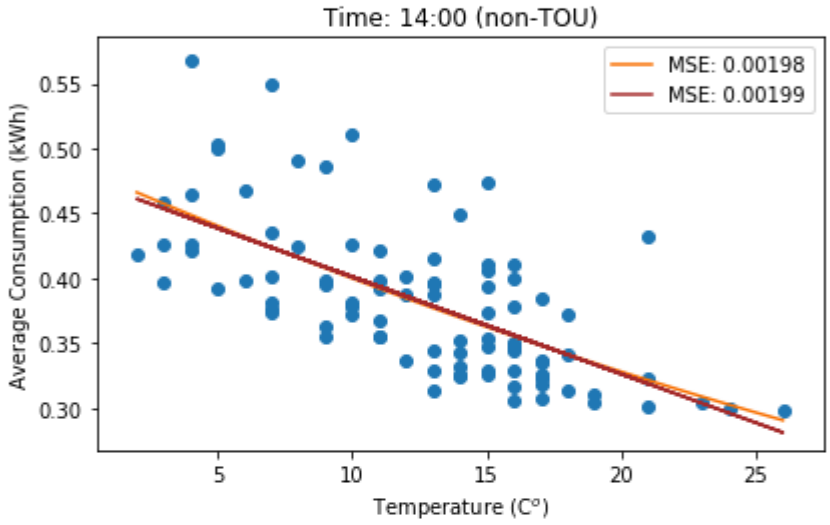

Average Consumption (kWh)
Temperature (C°)

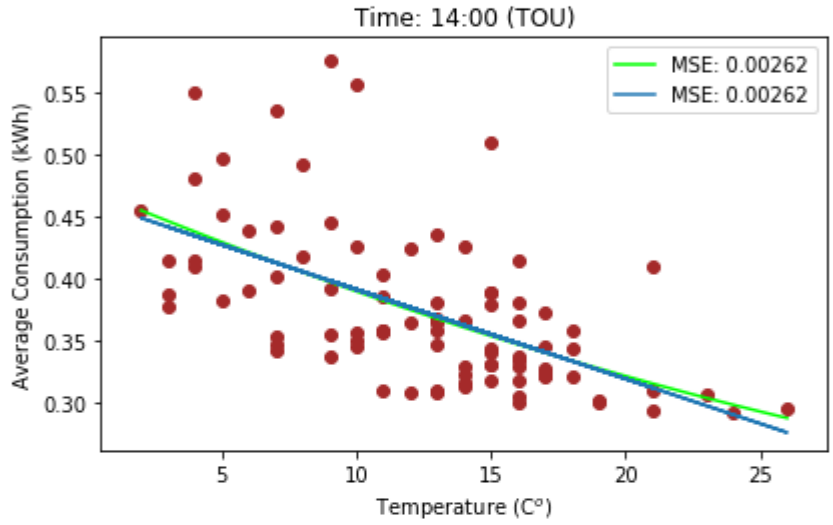

Time: 12:00 (non-TOU)
MSE: 0.00198
MSE: 0.00198
Average Consumption (kWh)
Temperature (C°)

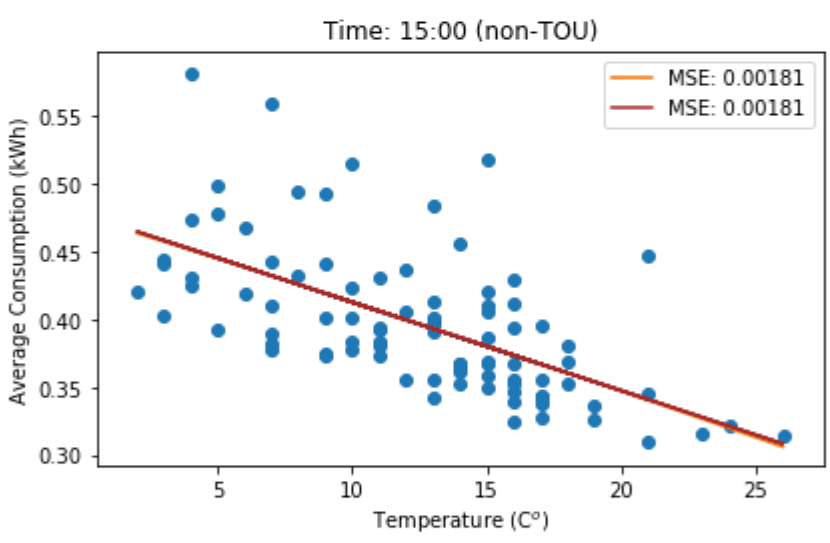

Time: 12:00 (TOU)
MSE: 0.00258
MSE: 0.00259
Average Consumption (kWh)
Temperature (C°)

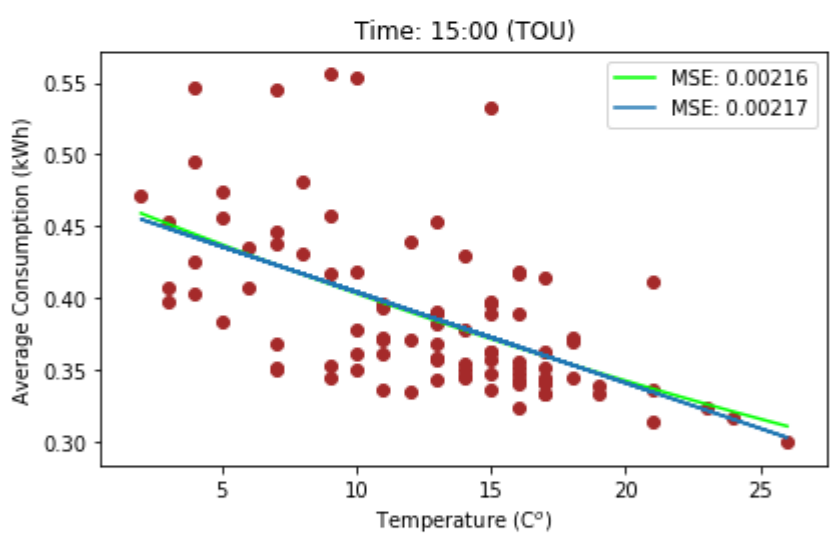

Time: 13:00 (non-TOU)
MSE: 0.00213
MSE: 0.00213
Average Consumption (kWh)
Temperature (C°)

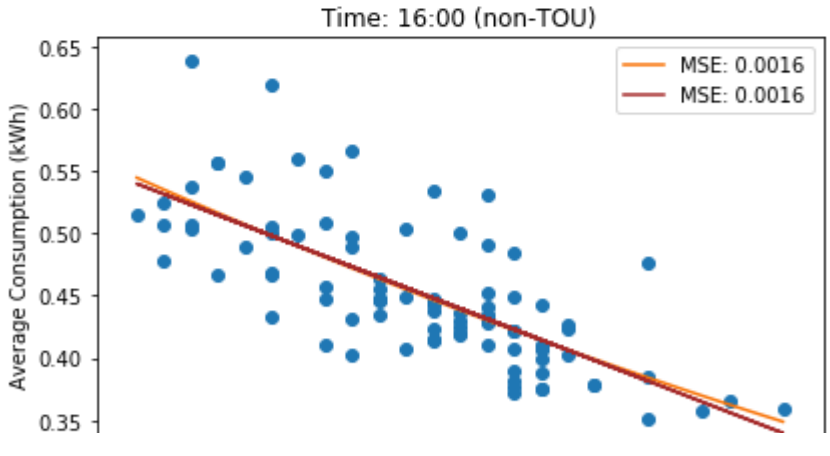

Time: 13:00 (TOU)
MSE: 0.00269
MSE: 0.00271
Average Consumption (kWh)
Temperature (C°)

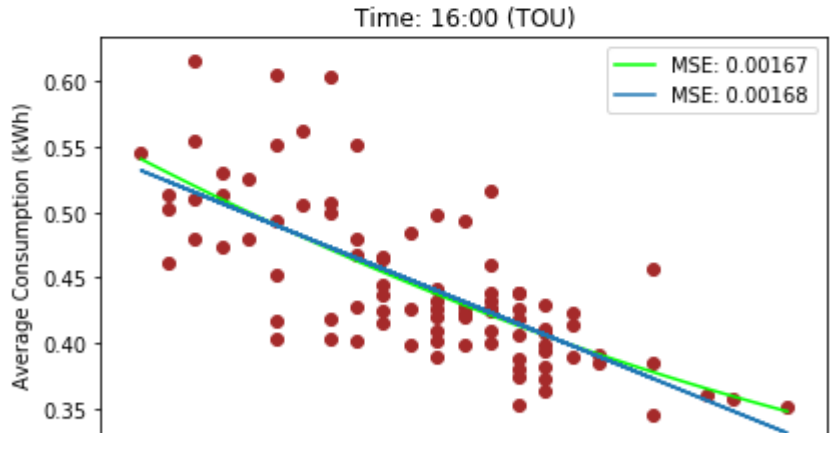

Time: 14:00 (non-TOU)
MSE: 0.00198
MSE: 0.00199
Average Consumption (kWh)
Temperature (C°)

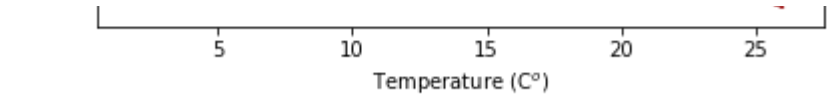

Time: 14:00 (TOU)
MSE: 0.00262
MSE: 0.00262
Average Consumption (kWh)
Temperature (C°)

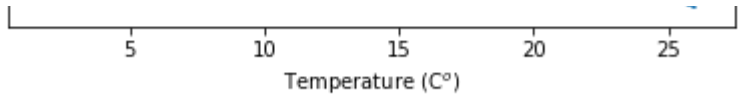

Time: 15:00 (non-TOU)
MSE: 0.00181
MSE: 0.00181
Average Consumption (kWh)
Temperature (C°)

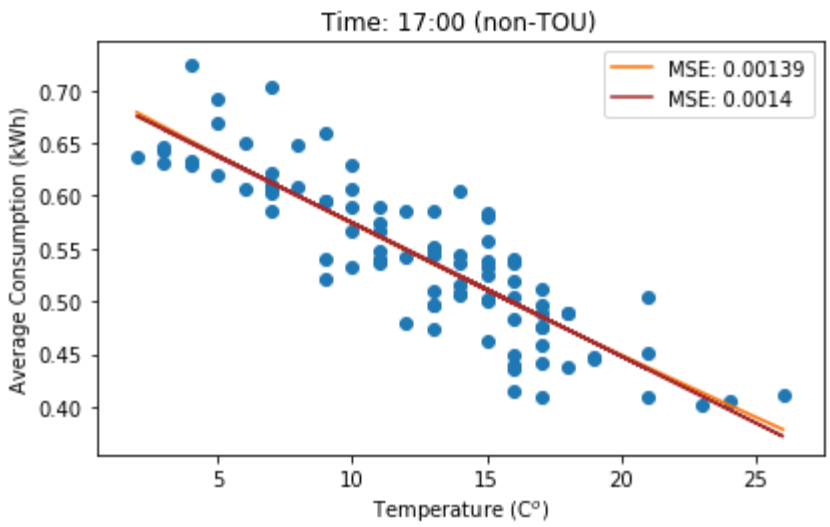

Time: 15:00 (TOU)
MSE: 0.00216
MSE: 0.00217
Average Consumption (kWh)
Temperature (C°)

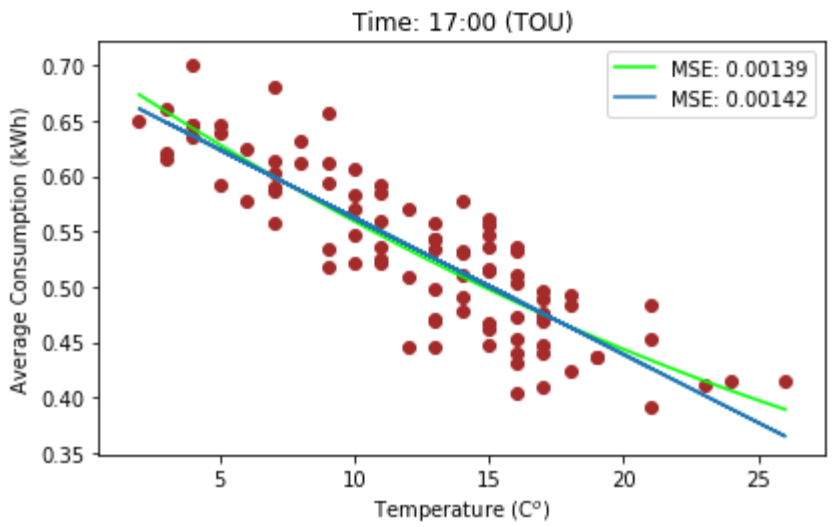

Time: 16:00 (non-TOU)
MSE: 0.0016
MSE: 0.0016
Average Consumption (kWh)

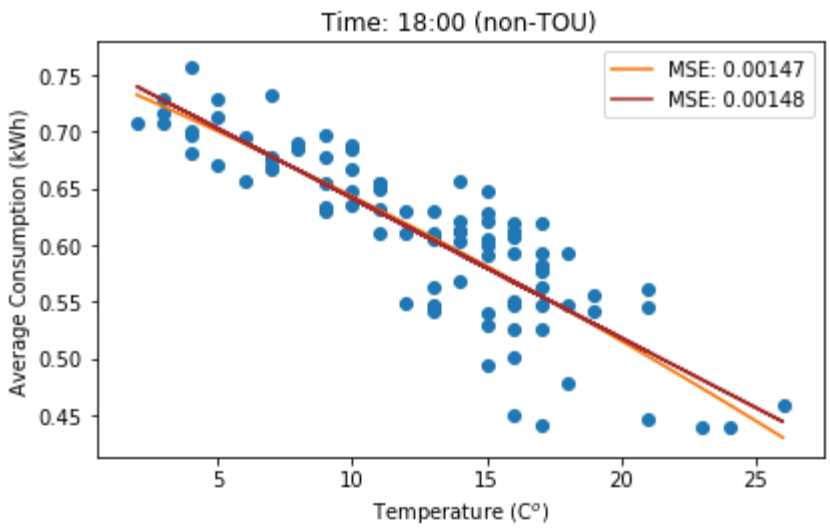

Time: 16:00 (TOU)
MSE: 0.00167
MSE: 0.00168
Average Consumption (kWh)

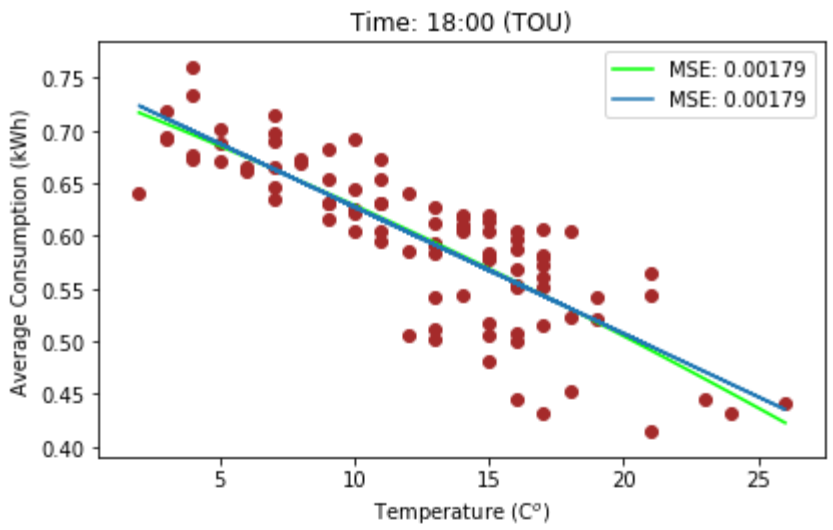

5
10
15
20
25
Temperature (C°)

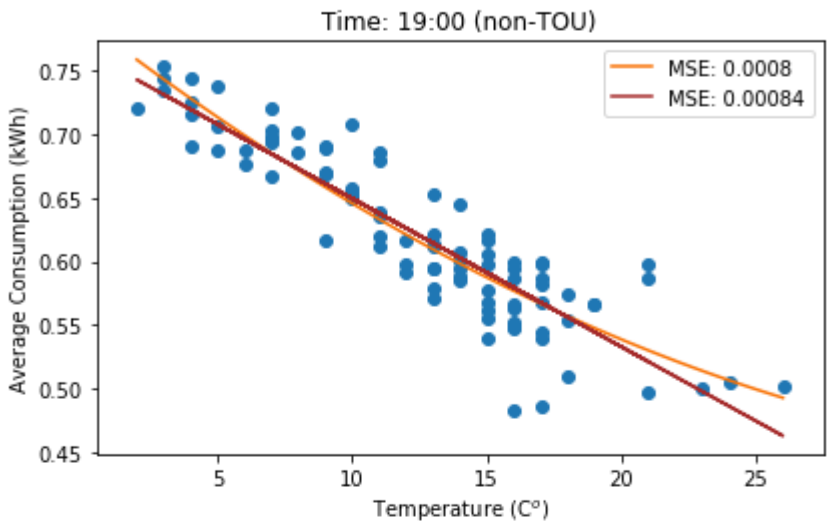

5
10
15
20
25
Temperature (C°)

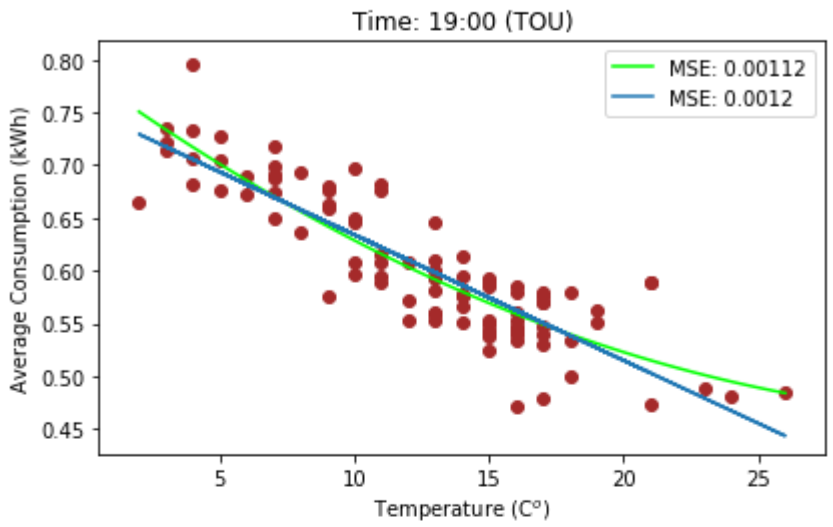

Time: 17:00 (non-TOU)
MSE: 0.00139
MSE: 0.0014
Average Consumption (kWh)
0.70
0.65
0.60
0.55
0.50
0.45
0.40
5
10
15
20
25
Temperature (C°)

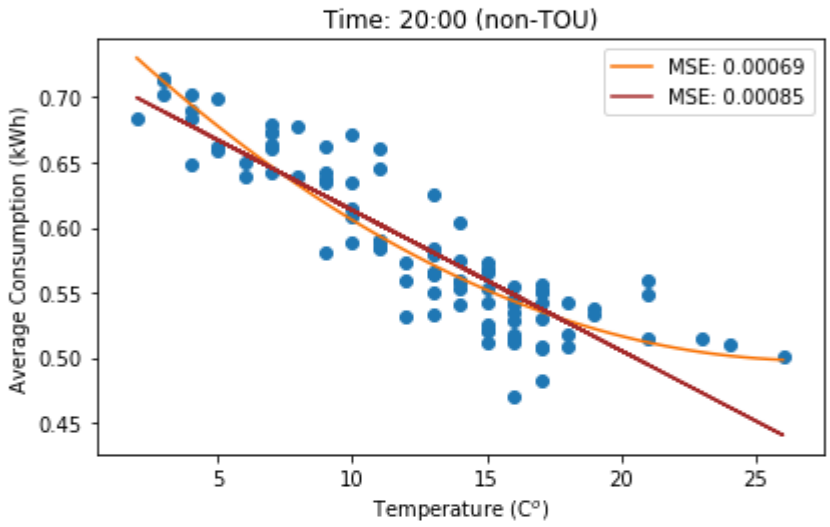

Time: 17:00 (TOU)
MSE: 0.00139
MSE: 0.00142
Average Consumption (kWh)
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
5
10
15
20
25
Temperature (C°)

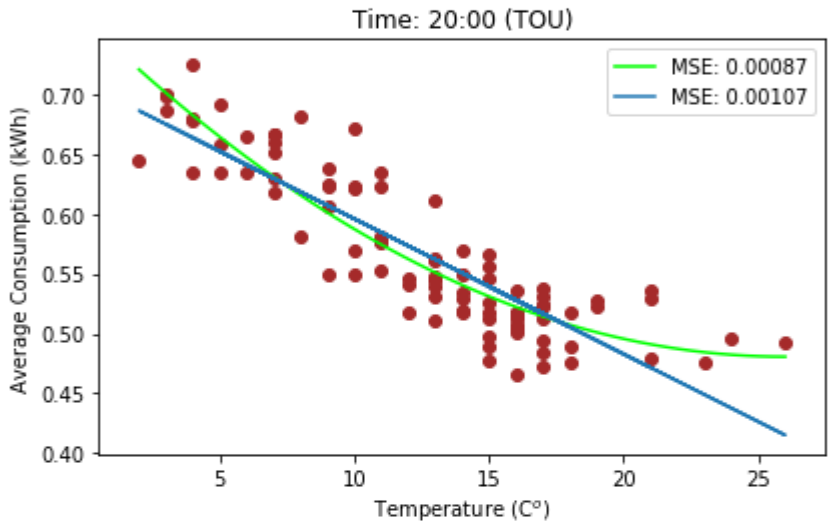

Time: 18:00 (non-TOU)
MSE: 0.00147
MSE: 0.00148
Average Consumption (kWh)
0.75
0.70
0.65
0.60
0.55
0.50
0.45
5
10
15
20
25
Temperature (C°)

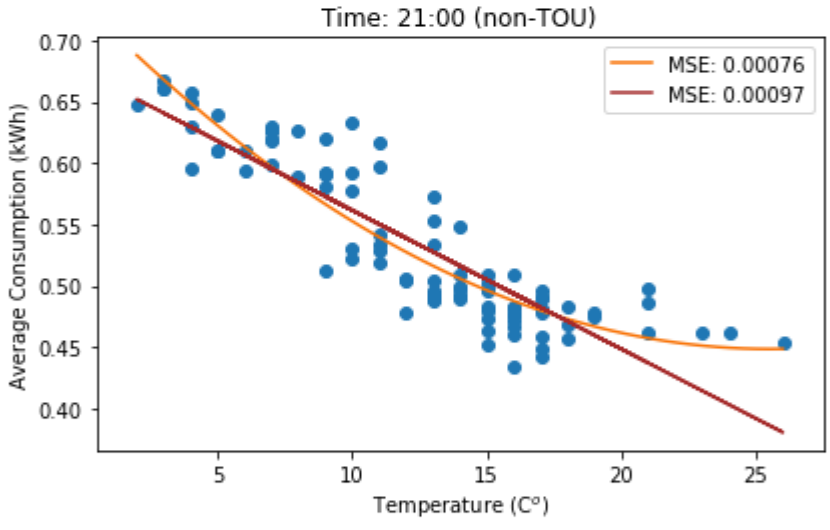

Time: 18:00 (TOU)
MSE: 0.00179
MSE: 0.00179
Average Consumption (kWh)
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
5
10
15
20
25
Temperature (C°)

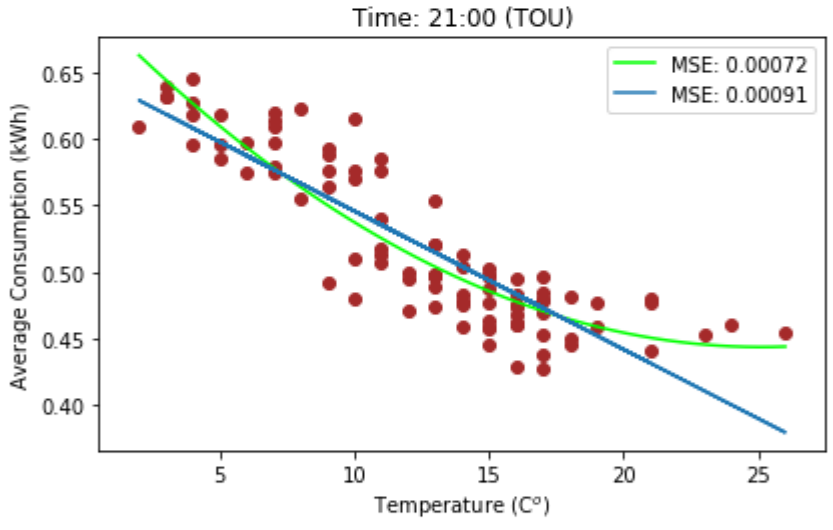

Time: 19:00 (non-TOU)
MSE: 0.0008
MSE: 0.00084
Average Consumption (kWh)
0.75
0.70
0.65
0.60
0.55
0.50
0.45
5
10
15
20
25
Temperature (C°)


Time: 19:00 (TOU)
MSE: 0.00112
MSE: 0.0012
Average Consumption (kWh)
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
5
10
15
20
25
Temperature (C°)


Time: 20:00 (non-TOU)
MSE: 0.00069
MSE: 0.00085
Average Consumption (kWh)
0.70
0.65
0.60
0.55
0.50
0.45
5
10
15
20
25
Temperature (C°)

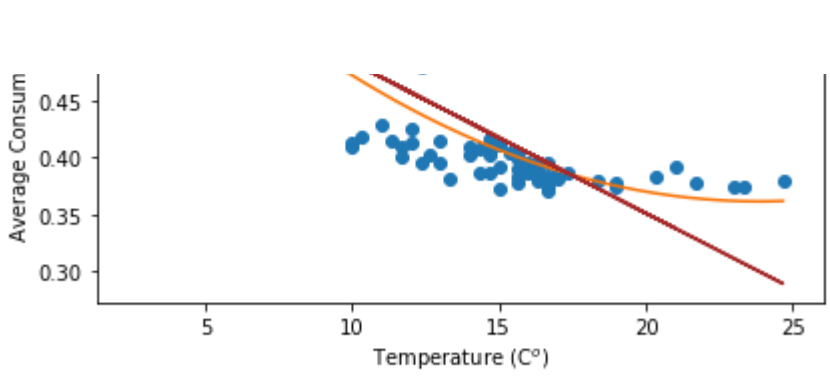

Time: 20:00 (TOU)
MSE: 0.00087
MSE: 0.00107
Average Consumption (kWh)
0.70
0.65
0.60
0.55
0.50
0.45
0.40
5
10
15
20
25
Temperature (C°)

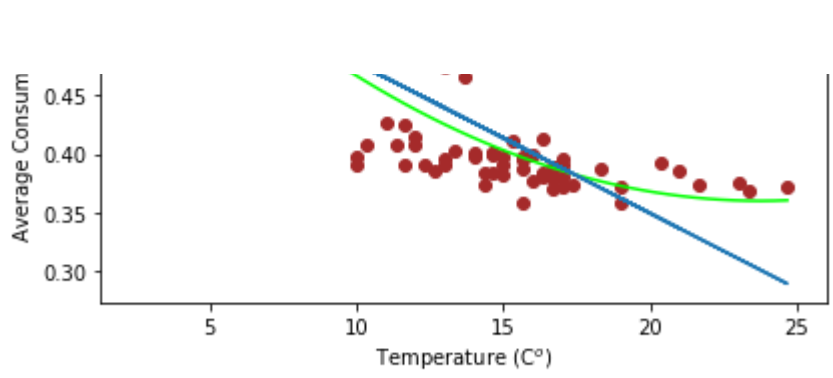

Time: 21:00 (non-TOU)
MSE: 0.00076
MSE: 0.00097
Average Consumption (kWh)
0.70
0.65
0.60
0.55
0.50
0.45
0.40
5
10
15
20
25
Temperature (C°)

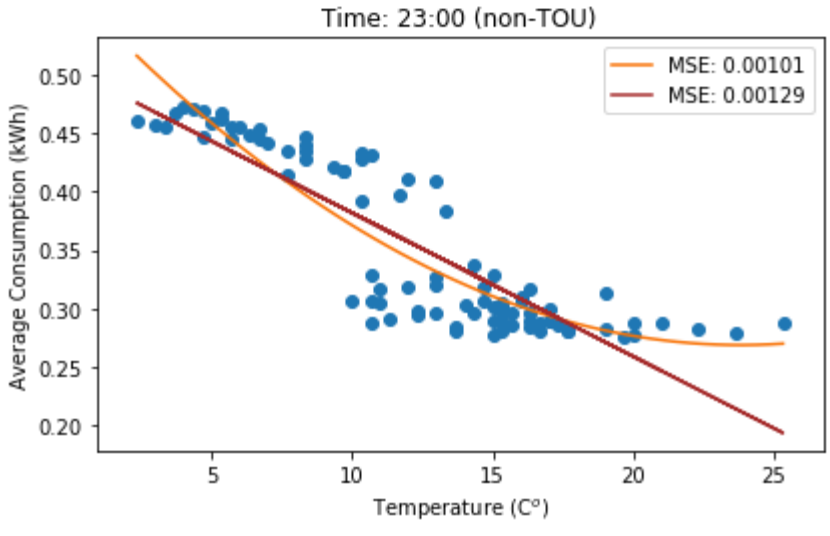

Time: 21:00 (TOU)
MSE: 0.00072
MSE: 0.00091
Average Consumption (kWh)
0.65
0.60
0.55
0.50
0.45
0.40
5
10
15
20
25
Temperature (C°)

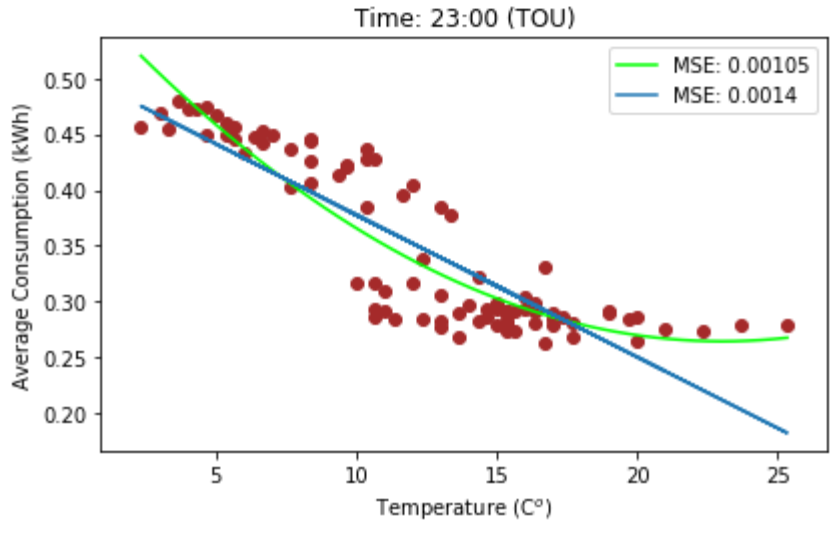
Time: 22:00 (non-TOU)
MSE: 0.0009
MSE: 0.00121
0.60
0.55
0.50

Time: 22:00 (TOU)
MSE: 0.00102
MSE: 0.00132
0.60
0.55
0.50

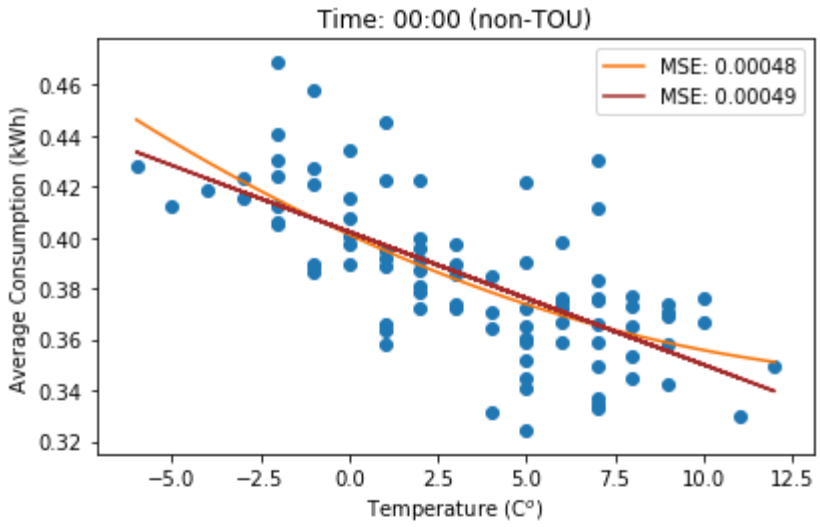

Average Consum
0.45
0.40
0.35
0.30
5
10
15
20
25
Temperature (C°)

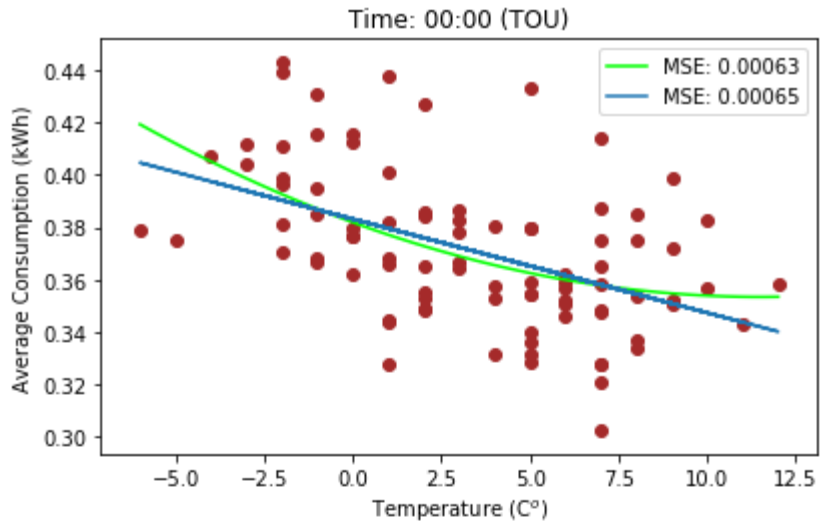

Average Consum
0.45
0.40
0.35
0.30
5
10
15
20
25
Temperature (C°)

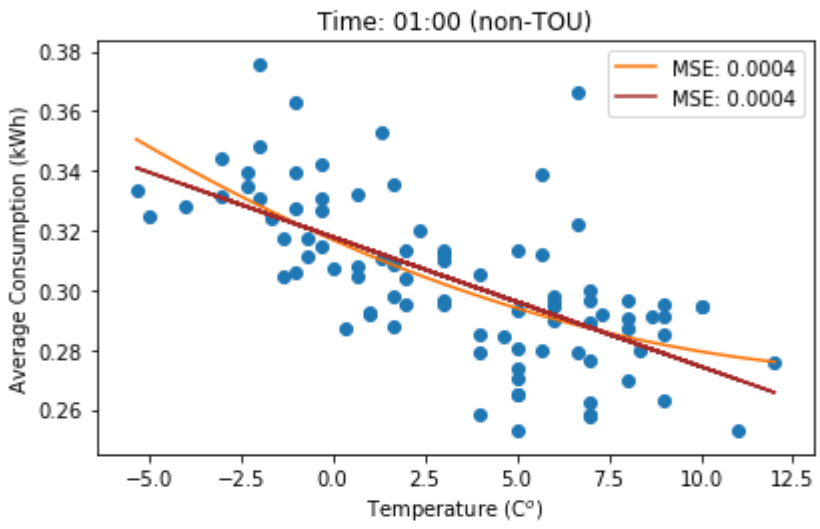

Time: 23:00 (non-TOU)
MSE: 0.00101
MSE: 0.00129
Average Consumption (kWh)
0.50
0.45
0.40
0.35
0.30
0.25
0.20
5
10
15
20
25
Temperature (C°)

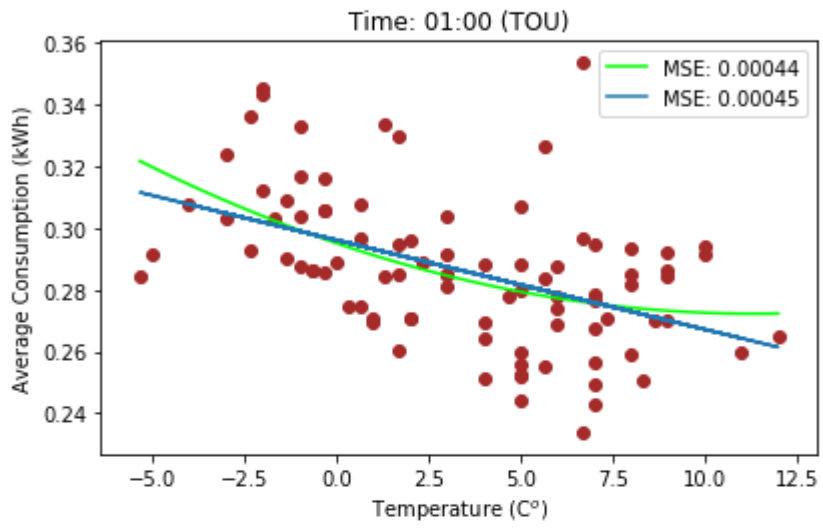

Time: 23:00 (TOU)
MSE: 0.00105
MSE: 0.0014
Average Consumption (kWh)
0.50
0.45
0.40
0.35
0.30
0.25
0.20
5
10
15
20
25
Temperature (C°)

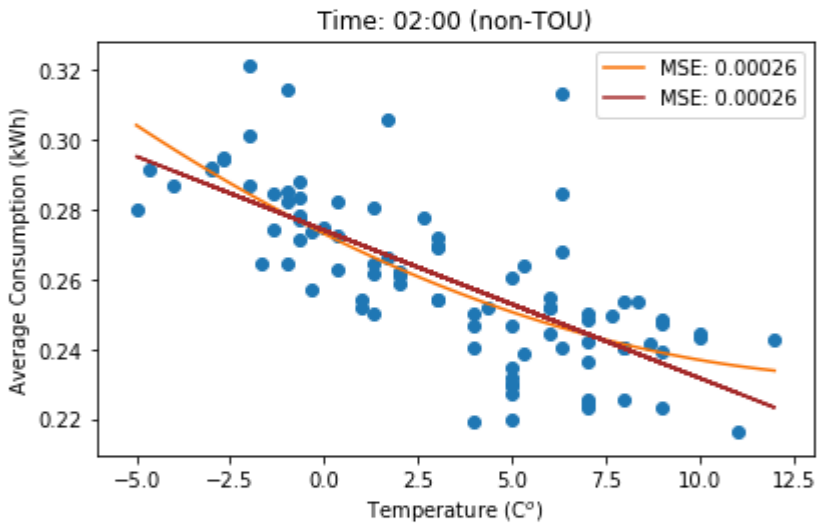
In [23]:
```
# Winter scatterplot of average consumption (over all consumers at same time period) vs. temp
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
# non-TOU vs TOU in terms of showing the elements in Average column such that they are hourly energy consumpt
ion v.s. temperature over 365 days
fig_all = plt.figure(figsize = (12,90))
ax_Ntou = [] # store subplot objects
ax_tou = [] # store subplot objects
lm = LinearRegression()
for i in range(24):
    ax_Ntou.append(fig_all.add_subplot(24, 2, 2 * i + 1))
    if i <= 9:
        x = df_wea1h_winter.TempC[df_wea1h_winter.GMT.str.contains('0' + str(i) + ':00:00')].values
        y = df_Ntou1h_winter.Average[df_Ntou1h_winter.GMT.str.contains('0' + str(i) + ':00:00')].values
        xp = np.linspace(min(x), max(x), 50) # polynomial regression
        z = np.polyfit(x, y, 2)
        p = np.poly1d(z)
        ax_Ntou[-1].plot(xp, p(xp), color = 'xkcd:orange', label = 'MSE: ' + str(round(mean_squared_error(y,
p(x)), 5)))
        x = x.reshape(-1, 1)
        y = y.reshape(-1, 1)
        lm.fit(x, y)
        ax_Ntou[-1].plot(x, lm.predict(x), color = 'brown', label = 'MSE: ' + str(round(mean_squared_error(y,
lm.predict(x)), 5)))
        ax_Ntou[-1].scatter(x, y)
        ax_Ntou[-1].set_title('Time: 0' + str(i) + ':00 (non-TOU)')
        ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_Ntou[-1].set_ylabel('Average Consumption (kWh)')
        ax_Ntou[-1].legend()
    else:
        x = df_wea1h_winter.TempC[df_wea1h_winter.GMT.str.contains(str(i) + ':00:00')].values
        y = df_Ntou1h_winter.Average[df_Ntou1h_winter.GMT.str.contains(str(i) + ':00:00')].values
        xp = np.linspace(min(x), max(x), 50) # polynomial regression
        z = np.polyfit(x, y, 2)
        p = np.poly1d(z)
        ax_Ntou[-1].plot(xp, p(xp), color = 'xkcd:orange', label = 'MSE: ' + str(round(mean_squared_error(y,
p(x)), 5)))
        x = x.reshape(-1, 1)
        y = y.reshape(-1, 1)
        lm.fit(x, y)
        ax_Ntou[-1].plot(x, lm.predict(x), color = 'brown', label = 'MSE: ' + str(round(mean_squared_error(y,
lm.predict(x)), 5)))
        ax_Ntou[-1].scatter(x, y)
        ax_Ntou[-1].set_title('Time: ' + str(i) + ':00 (non-TOU)')
        ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_Ntou[-1].set_ylabel('Average Consumption (kWh)')
        ax_Ntou[-1].legend()
for i in range(24):
    ax_tou.append(fig_all.add_subplot(24, 2, 2 * i + 2))
    if i <= 9:
        x = df_wea1h_winter.TempC[df_wea1h_winter.GMT.str.contains('0' + str(i) + ':00:00')].values
        y = df_tou1h_winter.Average[df_tou1h_winter.GMT.str.contains('0' + str(i) + ':00:00')].values
        xp = np.linspace(min(x), max(x), 50) # polynomial regression
        z = np.polyfit(x, y, 2)
        p = np.poly1d(z)
        ax_tou[-1].plot(xp, p(xp), color = 'lime', label = 'MSE: ' + str(round(mean_squared_error(y, p(x)), 5
)))
        x = x.reshape(-1, 1)
        y = y.reshape(-1, 1)
        lm.fit(x, y)
        ax_tou[-1].plot(x, lm.predict(x), label = 'MSE: ' + str(round(mean_squared_error(y, lm.predict(x)), 5
)))
        ax_tou[-1].scatter(x, y, color = 'brown')
        ax_tou[-1].set_title('Time: 0' + str(i) + ':00 (TOU)')
        ax_tou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_tou[-1].set_ylabel('Average Consumption (kWh)')
        ax_tou[-1].legend()
    else:
        x = df_wea1h_winter.TempC[df_wea1h_winter.GMT.str.contains(str(i) + ':00:00')].values
        y = df_tou1h_winter.Average[df_tou1h_winter.GMT.str.contains(str(i) + ':00:00')].values
        xp = np.linspace(min(x), max(x), 50) # polynomial regression
        z = np.polyfit(x, y, 2)
        p = np.poly1d(z)
        ax_tou[-1].plot(xp, p(xp), color = 'lime', label = 'MSE: ' + str(round(mean_squared_error(y, p(x)), 5
)))
        x = x.reshape(-1, 1)
        y = y.reshape(-1, 1)
        lm.fit(x, y)
        ax_tou[-1].plot(x, lm.predict(x), label = 'MSE: ' + str(round(mean_squared_error(y, lm.predict(x)), 5
)))
        ax_tou[-1].scatter(x, y, color = 'brown')
        ax_tou[-1].set_title('Time: ' + str(i) + ':00 (TOU)')
        ax_tou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_tou[-1].set_ylabel('Average Consumption (kWh)')
        ax_tou[-1].legend()
plt.tight_layout()
```

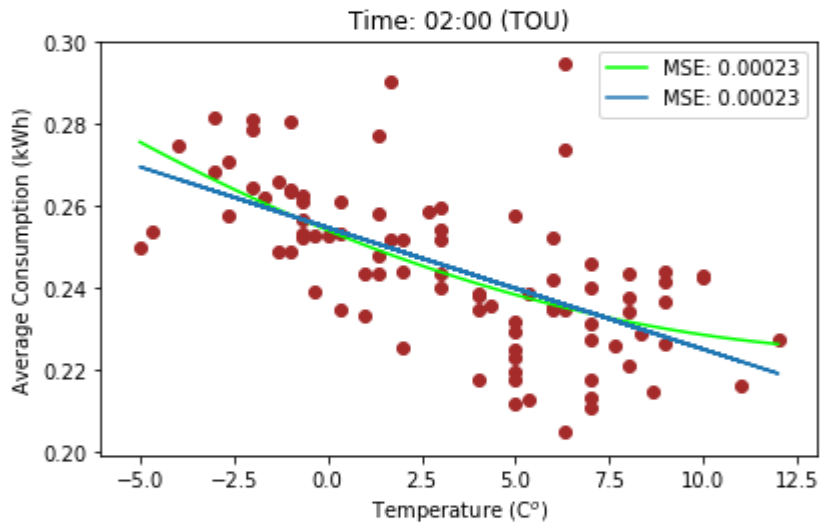

Time: 00:00 (non-TOU)
MSE: 0.00048
MSE: 0.00049
Average Consumption (kWh)
Temperature (C°)

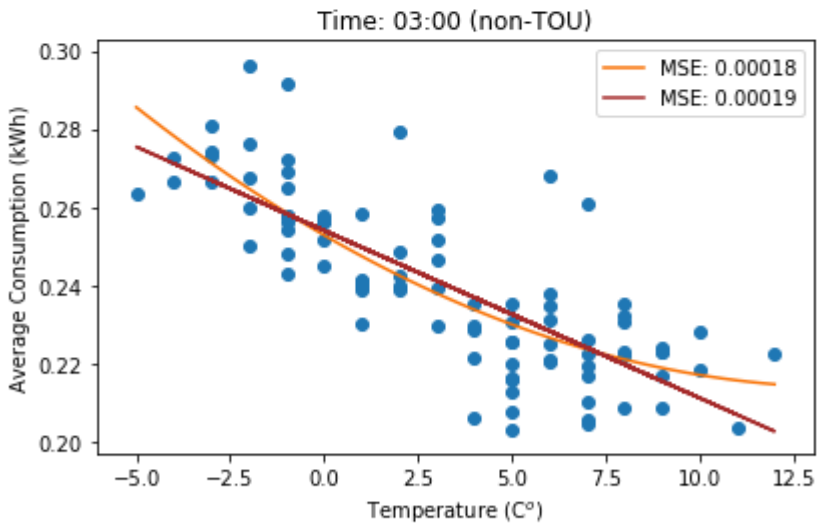

Time: 00:00 (TOU)
MSE: 0.00063
MSE: 0.00065
Average Consumption (kWh)
Temperature (C°)

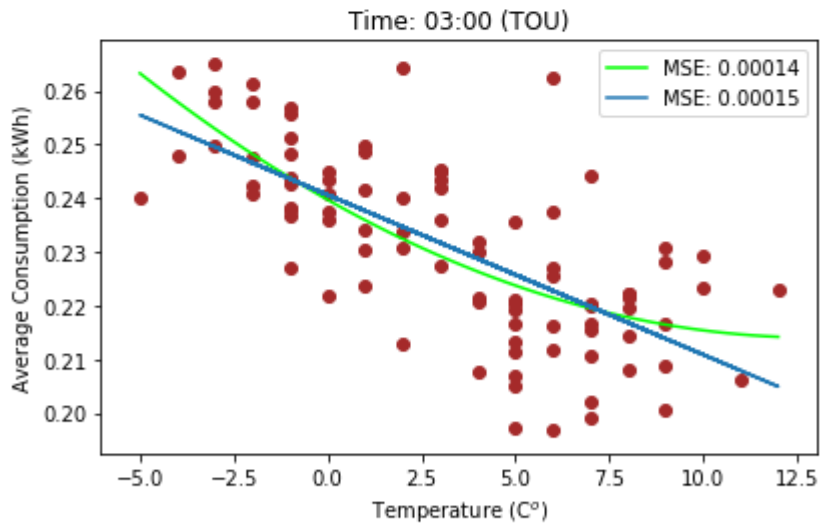

Time: 01:00 (non-TOU)
MSE: 0.0004
MSE: 0.0004
Average Consumption (kWh)
Temperature (C°)

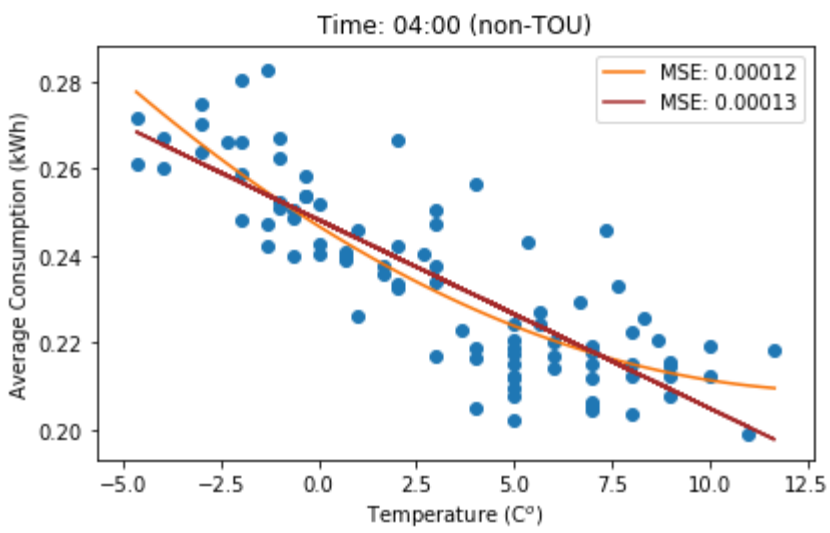

Time: 01:00 (TOU)
MSE: 0.00044
MSE: 0.00045
Average Consumption (kWh)
Temperature (C°)

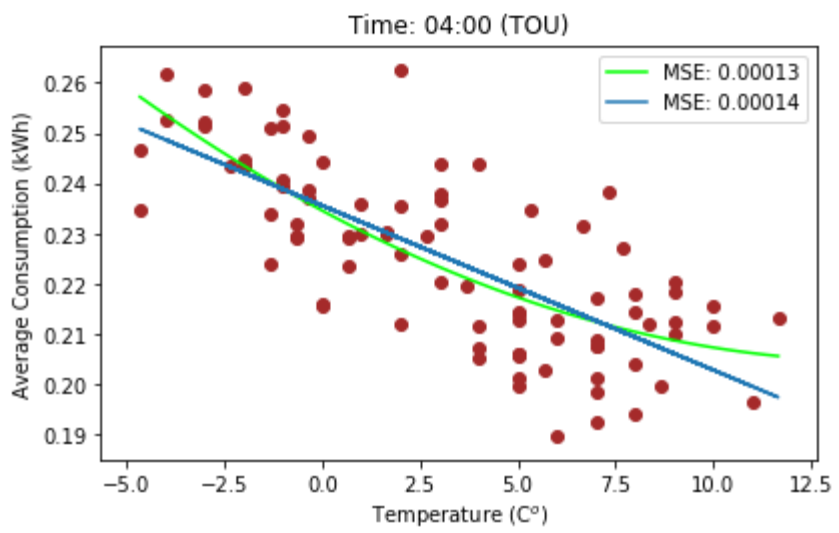

Time: 02:00 (non-TOU)
MSE: 0.00026
MSE: 0.00026
Average Consumption (kWh)
Temperature (C°)

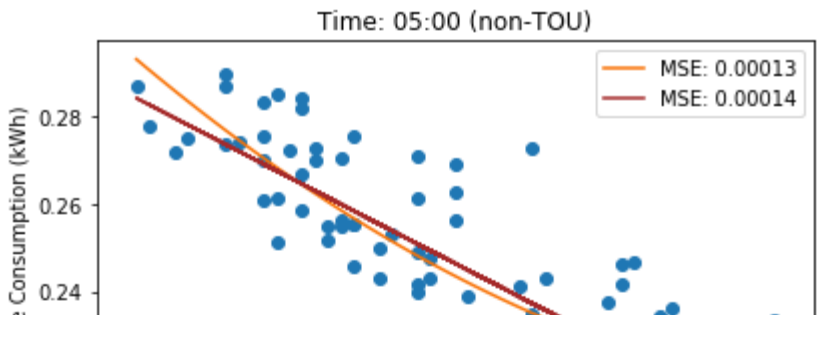

Time: 02:00 (TOU)
MSE: 0.00023
MSE: 0.00023
Average Consumption (kWh)
Temperature (C°)

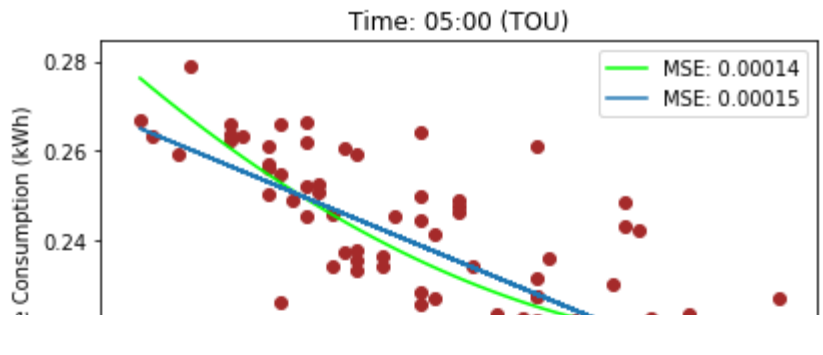

Time: 03:00 (non-TOU)
MSE: 0.00018
MSE: 0.00019
Average Consumption (kWh)
Temperature (C°)

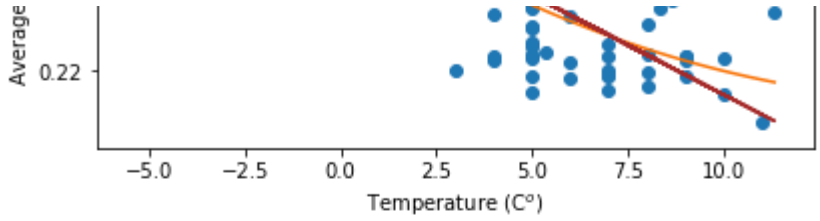

Time: 03:00 (TOU)
MSE: 0.00014
MSE: 0.00015
Average Consumption (kWh)
Temperature (C°)

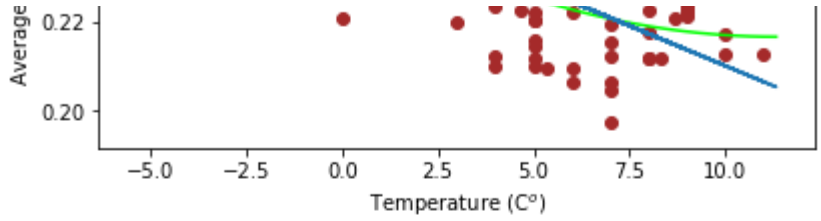

Time: 04:00 (non-TOU)
MSE: 0.00012
MSE: 0.00013
Average Consumption (kWh)
Temperature (C°)

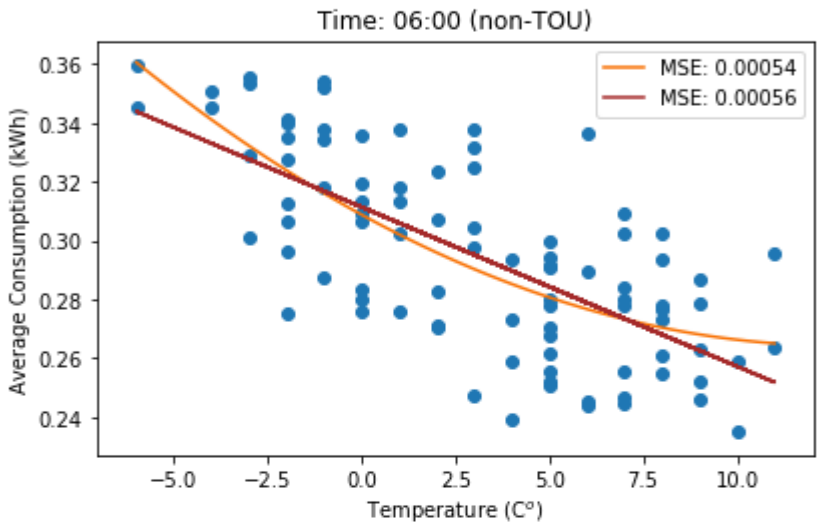

Time: 04:00 (TOU)
MSE: 0.00013
MSE: 0.00014
Average Consumption (kWh)
Temperature (C°)

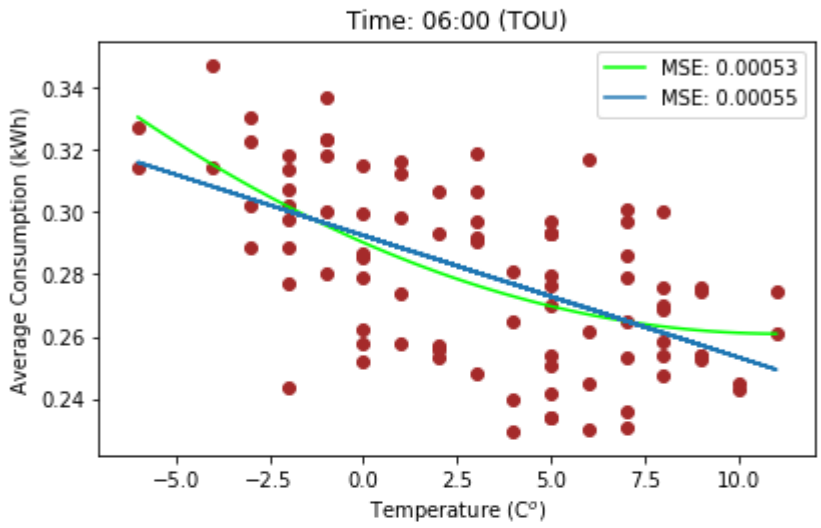

Time: 05:00 (non-TOU)
MSE: 0.00013
MSE: 0.00014
Average Consumption (kWh)

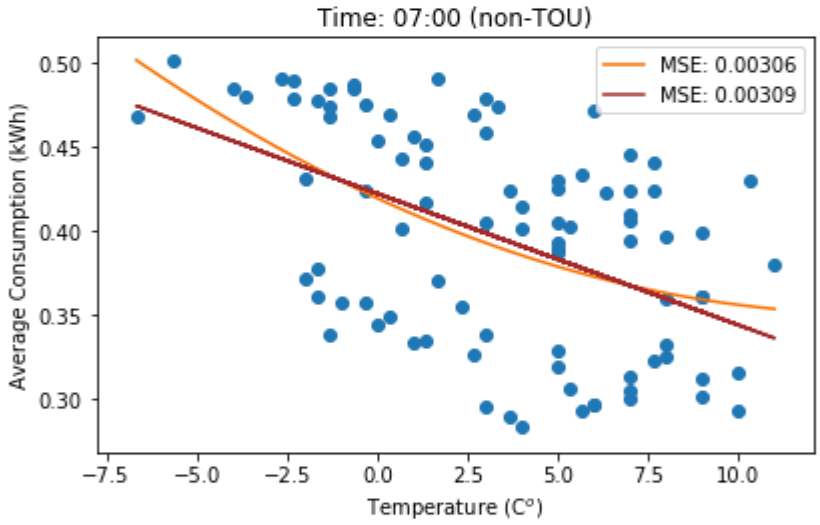

Time: 05:00 (TOU)
MSE: 0.00014
MSE: 0.00015
Average Consumption (kWh)

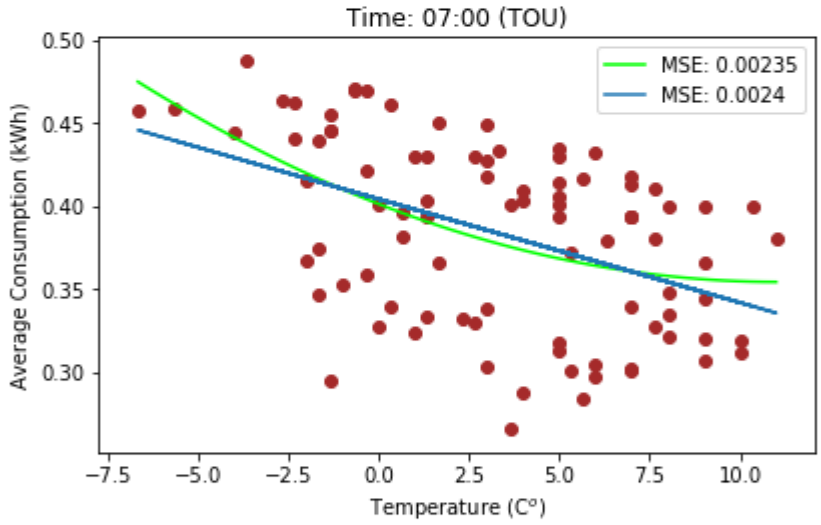

Average
0.22
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
Temperature (C°)

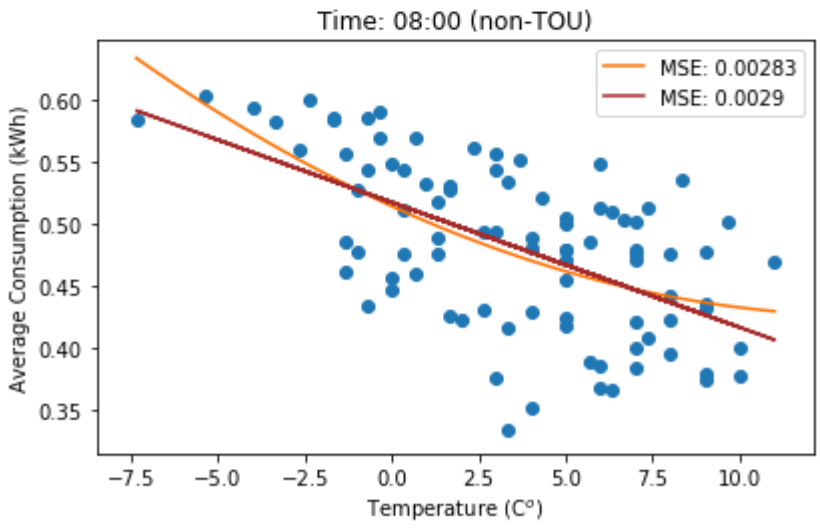

Average
0.22
0.20
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
Temperature (C°)

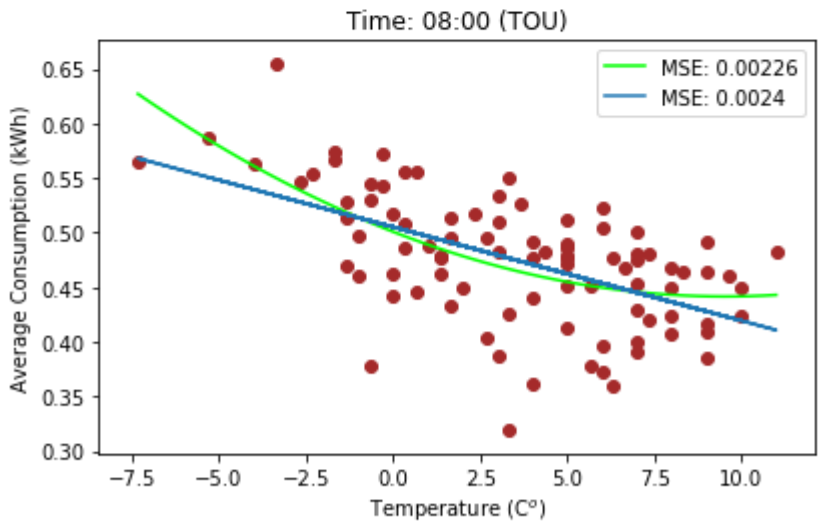

Time: 06:00 (non-TOU)
MSE: 0.00054
MSE: 0.00056
Average Consumption (kWh)
0.36
0.34
0.32
0.30
0.28
0.26
0.24
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
Temperature (C°)

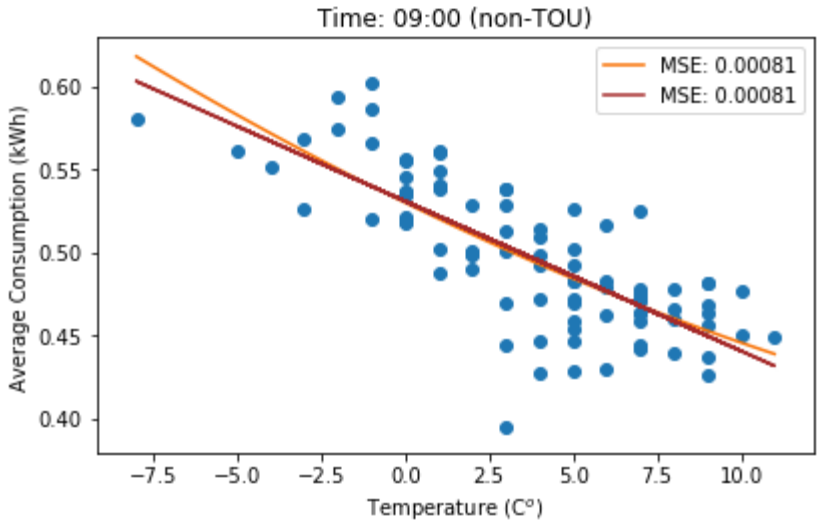

Time: 06:00 (TOU)
MSE: 0.00053
MSE: 0.00055
Average Consumption (kWh)
0.34
0.32
0.30
0.28
0.26
0.24
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
Temperature (C°)

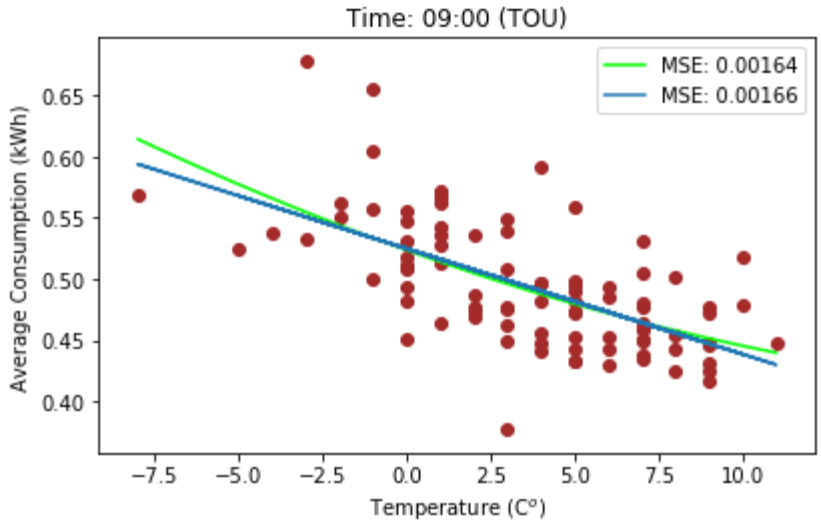

Time: 07:00 (non-TOU)
MSE: 0.00306
MSE: 0.00309
Average Consumption (kWh)
0.50
0.45
0.40
0.35
0.30
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
Temperature (C°)

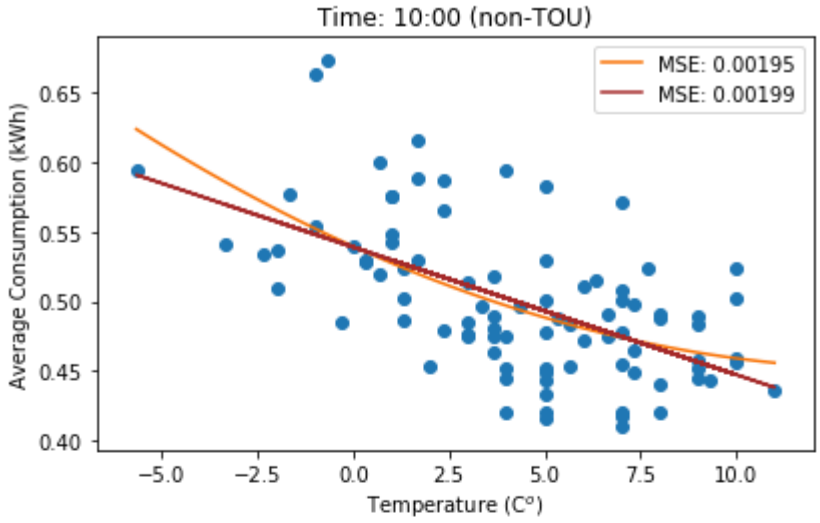

Time: 07:00 (TOU)
MSE: 0.00235
MSE: 0.0024
Average Consumption (kWh)
0.50
0.45
0.40
0.35
0.30
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
Temperature (C°)

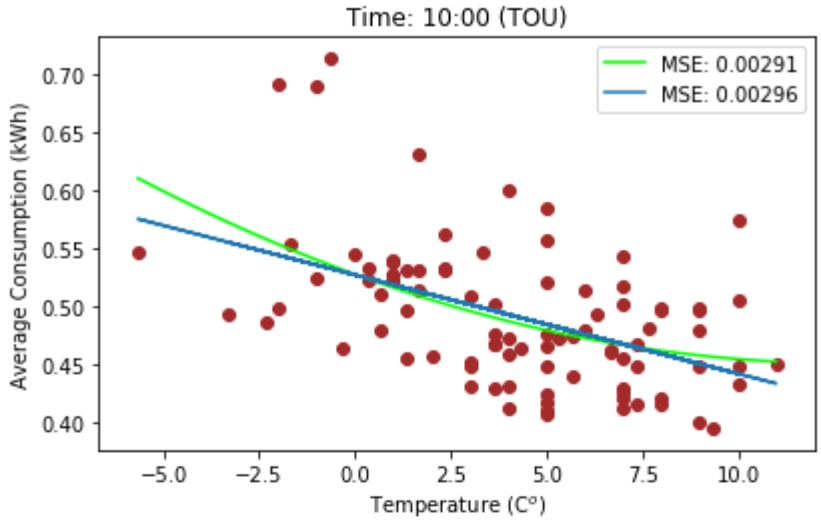

Time: 08:00 (non-TOU)
MSE: 0.00283
MSE: 0.0029
Average Consumption (kWh)
0.60
0.55
0.50
0.45
0.40
0.35
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
Temperature (C°)

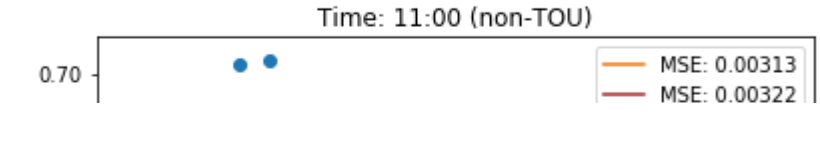

Time: 08:00 (TOU)
MSE: 0.00226
MSE: 0.0024
Average Consumption (kWh)
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
Temperature (C°)

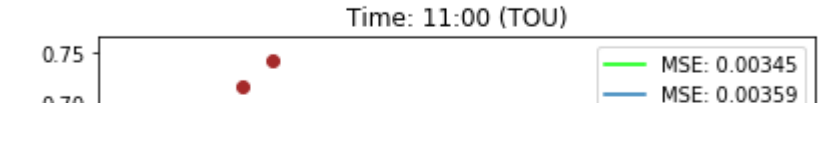

Time: 09:00 (non-TOU)
MSE: 0.00081
MSE: 0.00081
Average Consumption (kWh)
0.60
0.55
0.50
0.45
0.40
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
Temperature (C°)

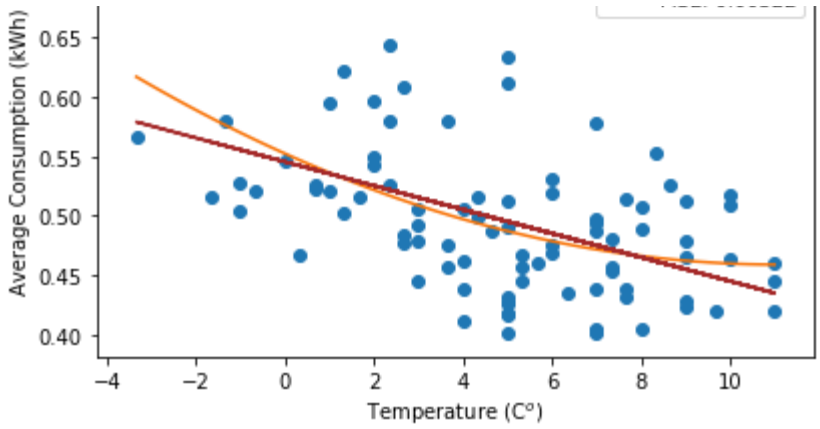

Time: 09:00 (TOU)
MSE: 0.00164
MSE: 0.00166
Average Consumption (kWh)
0.65
0.60
0.55
0.50
0.45
0.40
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
Temperature (C°)

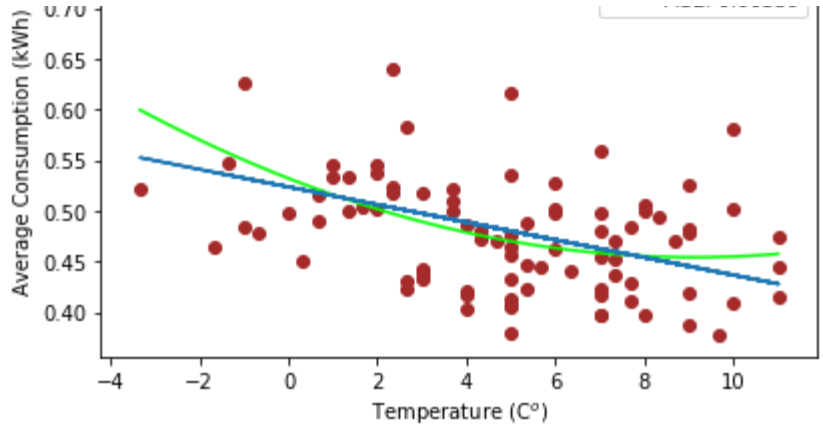

Time: 10:00 (non-TOU)
MSE: 0.00195
MSE: 0.00199
Average Consumption (kWh)
0.65
0.60
0.55
0.50
0.45
0.40
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
Temperature (C°)

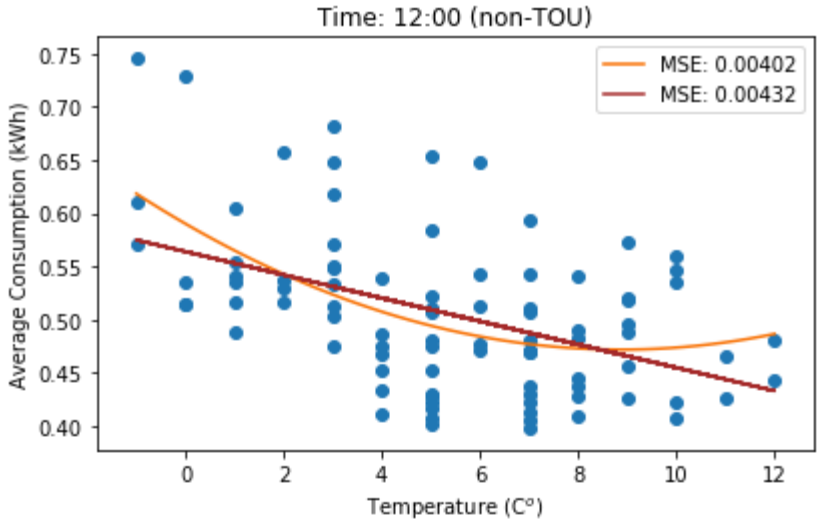

Time: 10:00 (TOU)
MSE: 0.00291
MSE: 0.00296
Average Consumption (kWh)
0.70
0.65
0.60
0.55
0.50
0.45
0.40
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
Temperature (C°)

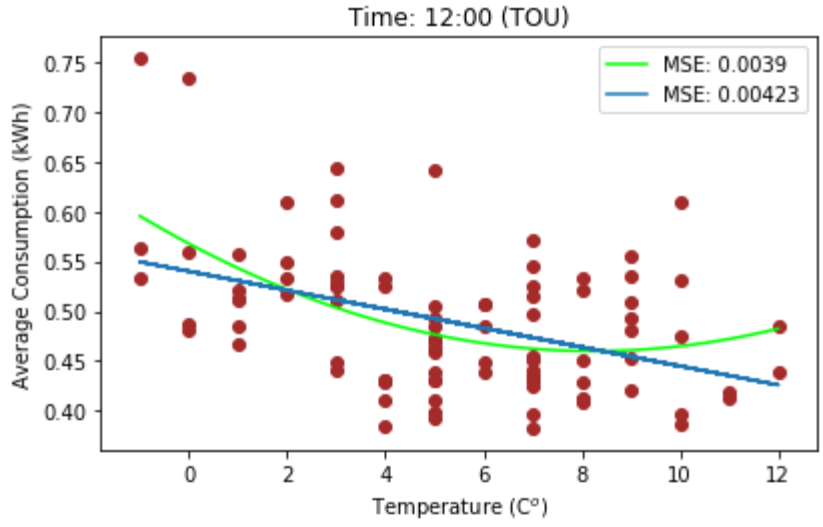

Time: 11:00 (non-TOU)
MSE: 0.00313
MSE: 0.00322
0.70

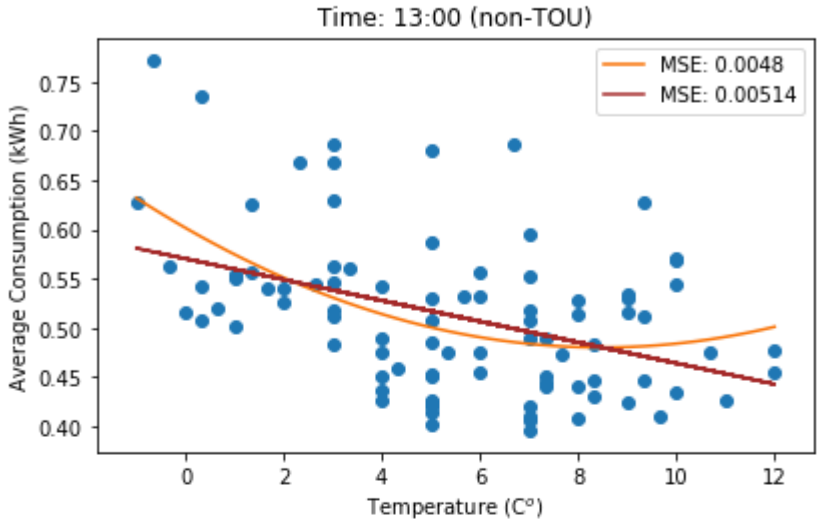

Time: 11:00 (TOU)
MSE: 0.00345
MSE: 0.00359
0.75

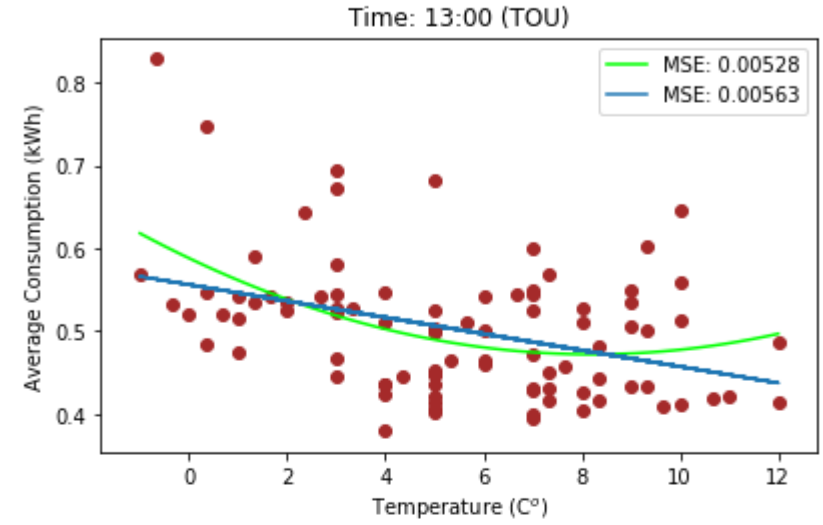

Average Consumption (kWh)
Temperature (C°)

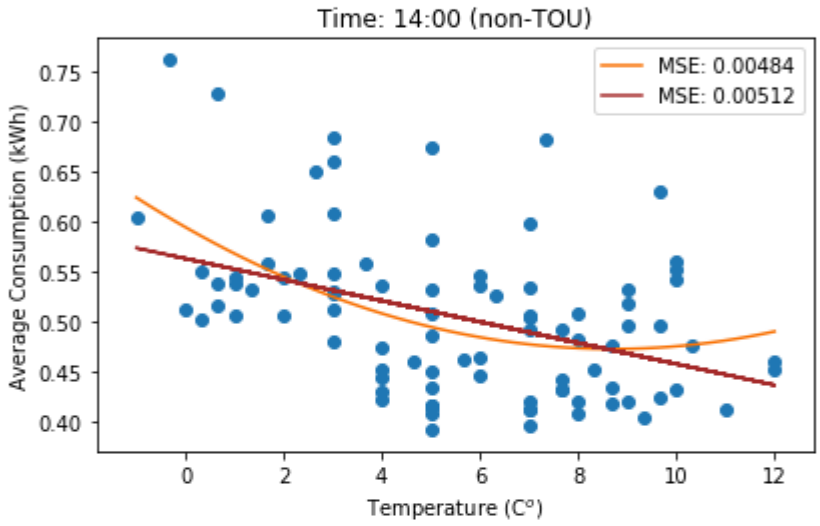

Average Consumption (kWh)
Temperature (C°)

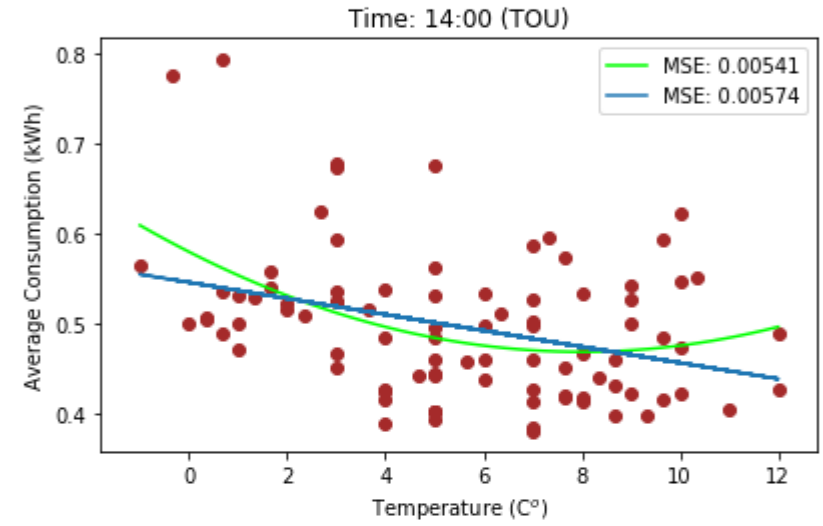

Time: 12:00 (non-TOU)
MSE: 0.00402
MSE: 0.00432
Average Consumption (kWh)
Temperature (C°)

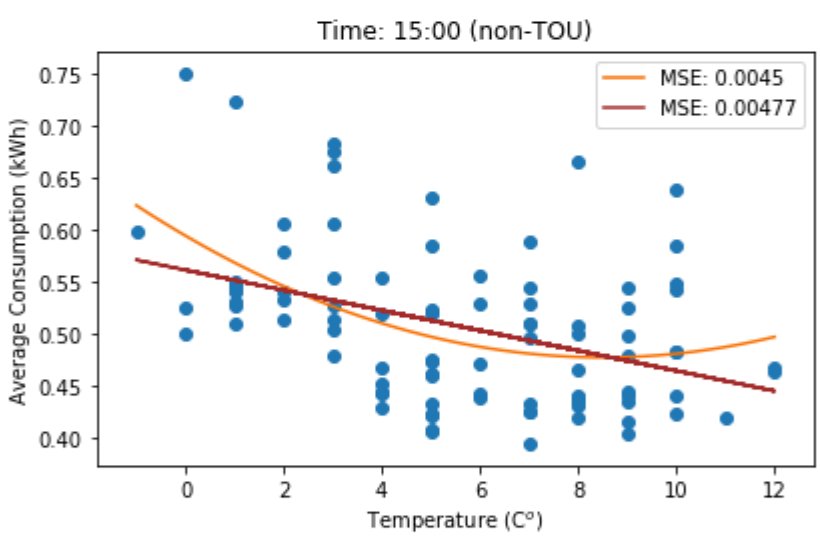

Time: 12:00 (TOU)
MSE: 0.0039
MSE: 0.00423
Average Consumption (kWh)
Temperature (C°)

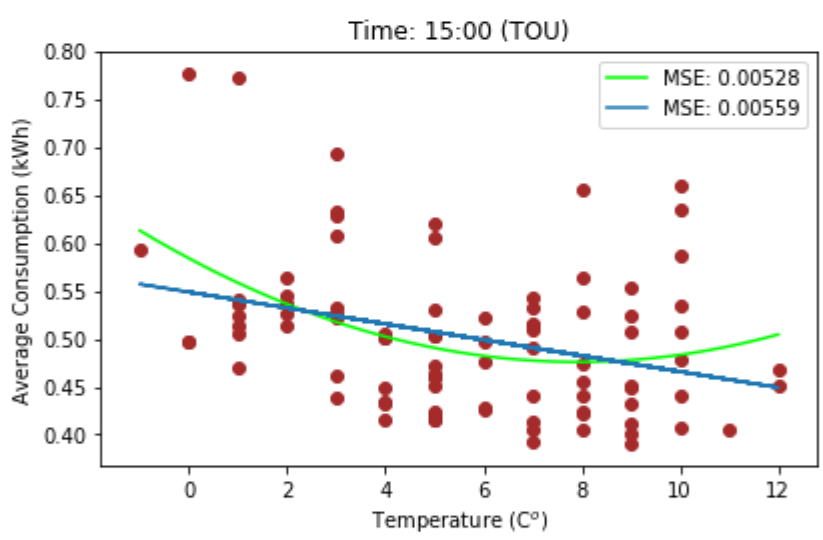

Time: 13:00 (non-TOU)
MSE: 0.0048
MSE: 0.00514
Average Consumption (kWh)
Temperature (C°)

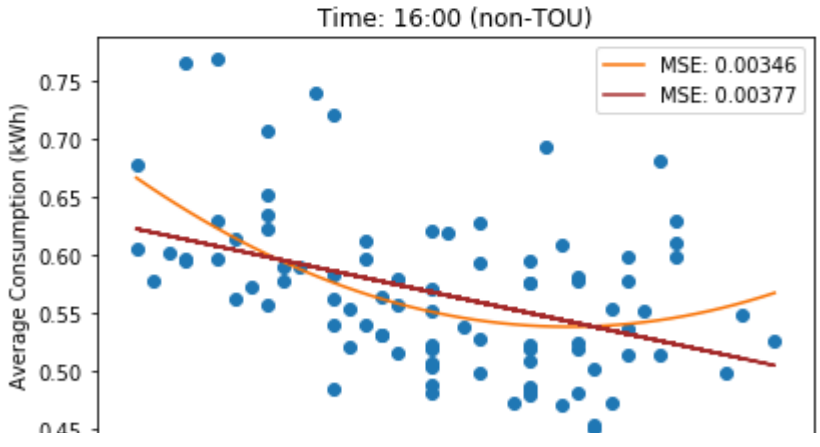

Time: 13:00 (TOU)
MSE: 0.00528
MSE: 0.00563
Average Consumption (kWh)
Temperature (C°)

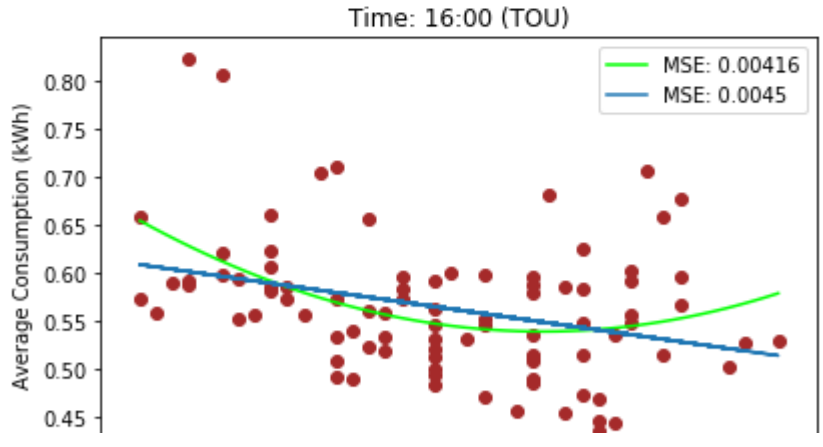

Time: 14:00 (non-TOU)
MSE: 0.00484
MSE: 0.00512
Average Consumption (kWh)
Temperature (C°)

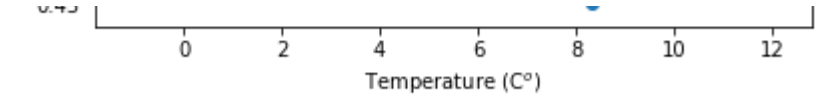

Time: 14:00 (TOU)
MSE: 0.00541
MSE: 0.00574
Average Consumption (kWh)
Temperature (C°)

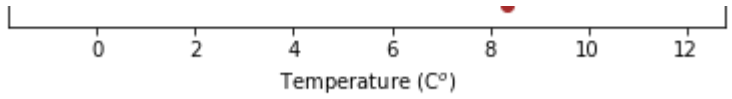

Time: 15:00 (non-TOU)
MSE: 0.0045
MSE: 0.00477
Average Consumption (kWh)
Temperature (C°)

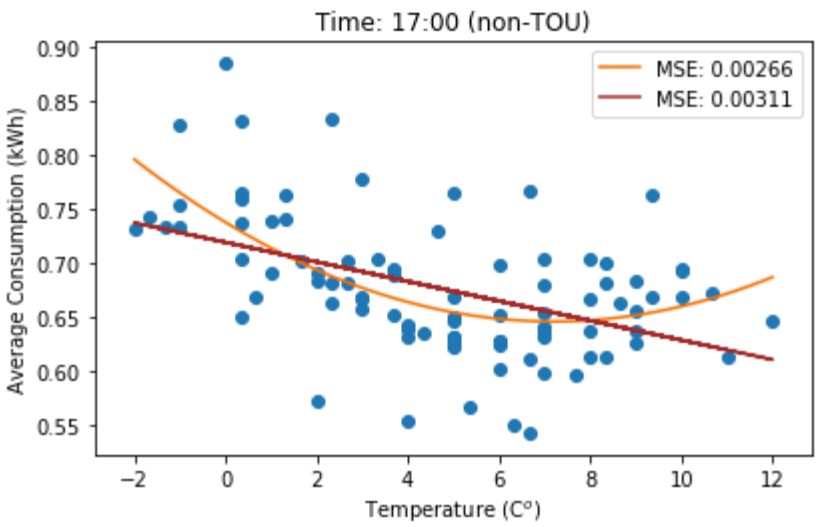

Time: 15:00 (TOU)
MSE: 0.00528
MSE: 0.00559
Average Consumption (kWh)
Temperature (C°)

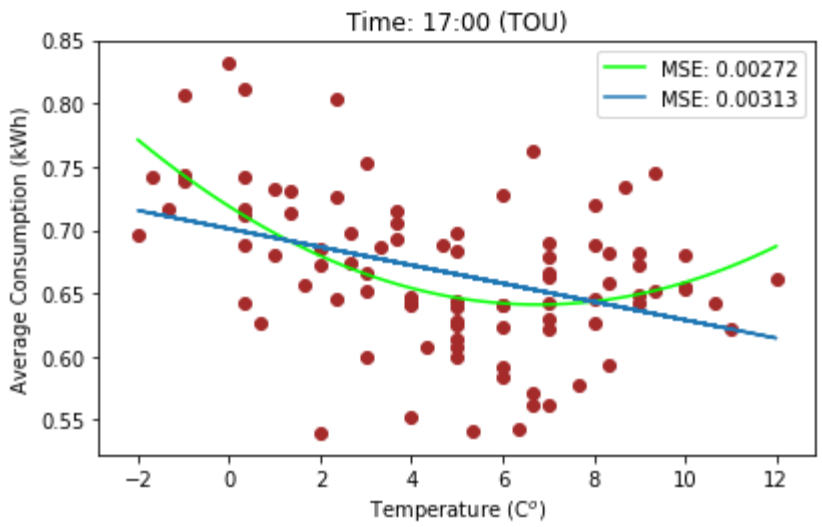

Time: 16:00 (non-TOU)
MSE: 0.00346
MSE: 0.00377
Average Consumption (kWh)

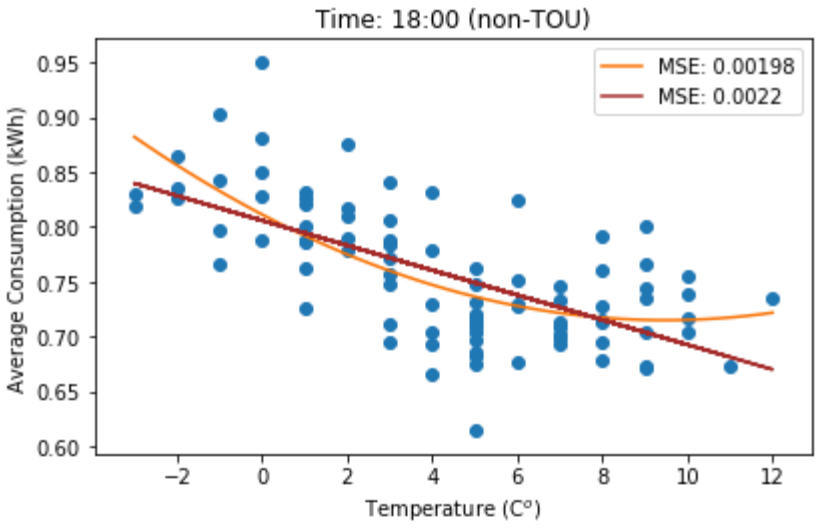

Time: 16:00 (TOU)
MSE: 0.00416
MSE: 0.0045
Average Consumption (kWh)

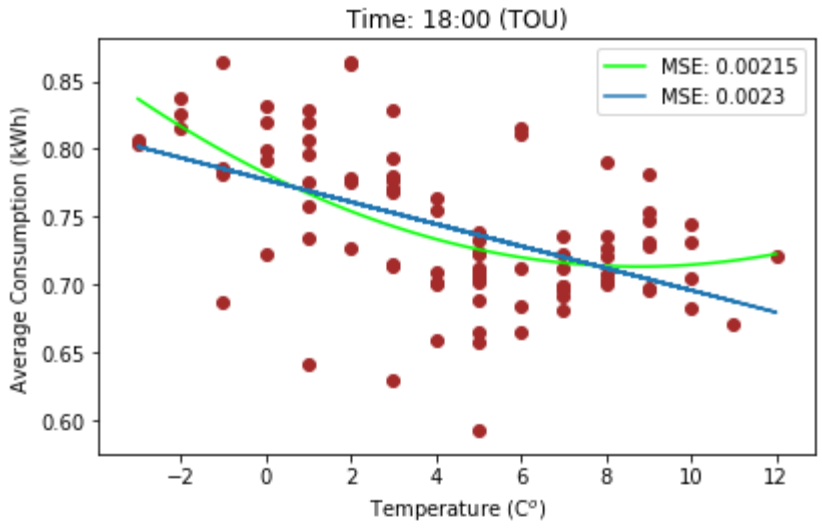

0
2
4
6
8
10
12
Temperature (C°)

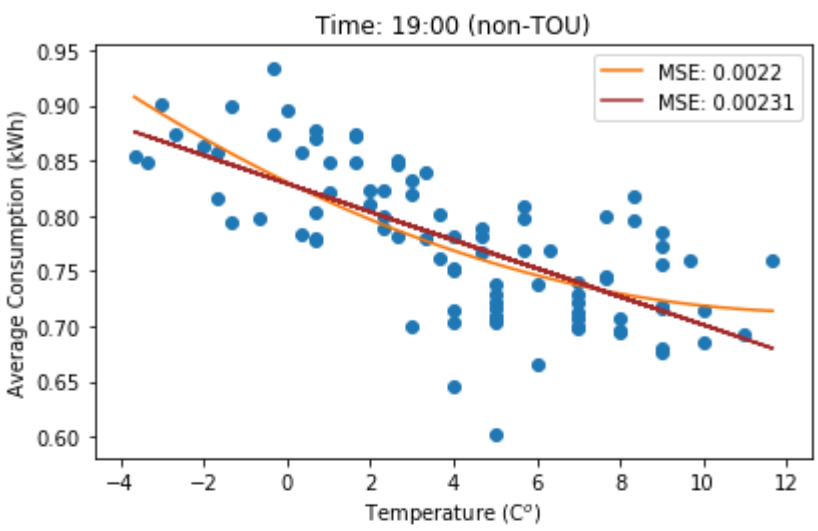

0
2
4
6
8
10
12
Temperature (C°)

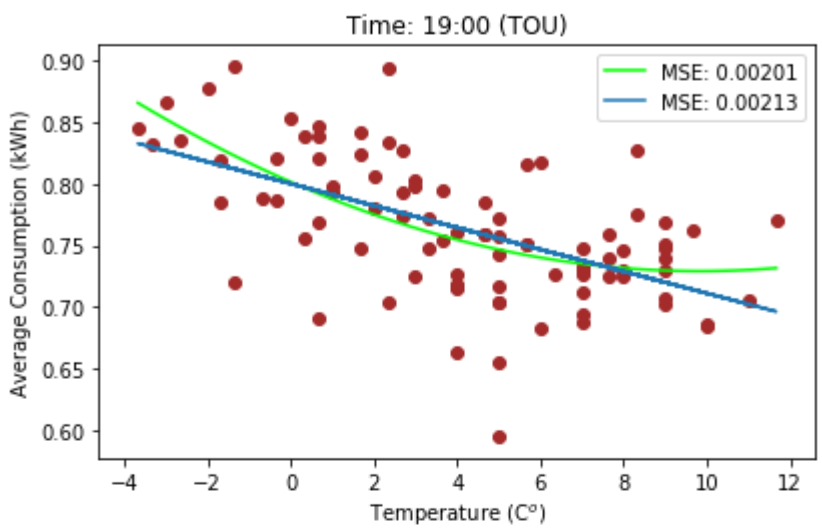

Time: 17:00 (non-TOU)
MSE: 0.00266
MSE: 0.00311
Average Consumption (kWh)
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
−2
0
2
4
6
8
10
12
Temperature (C°)

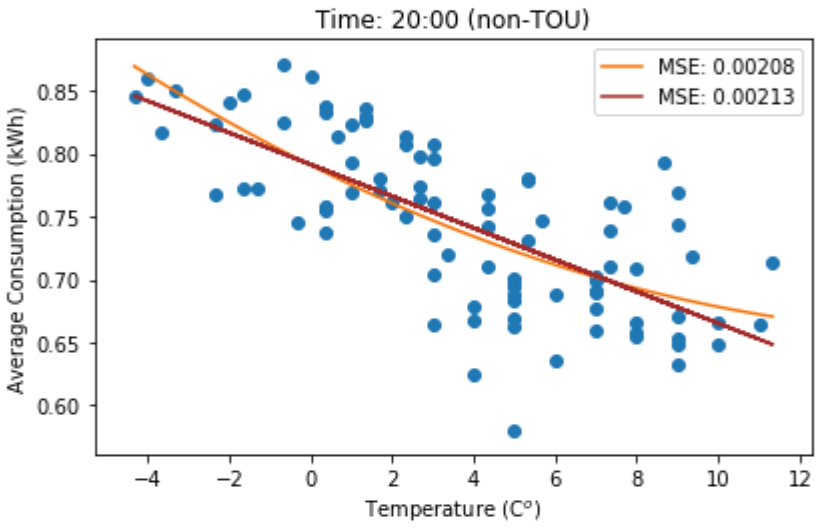

Time: 17:00 (TOU)
MSE: 0.00272
MSE: 0.00313
Average Consumption (kWh)
0.85
0.80
0.75
0.70
0.65
0.60
0.55
−2
0
2
4
6
8
10
12
Temperature (C°)

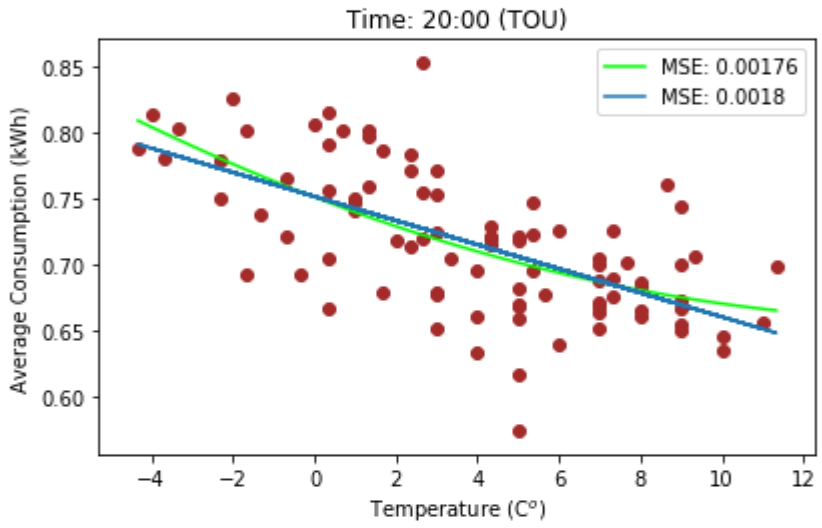

Time: 18:00 (non-TOU)
MSE: 0.00198
MSE: 0.0022
Average Consumption (kWh)
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
−2
0
2
4
6
8
10
12
Temperature (C°)

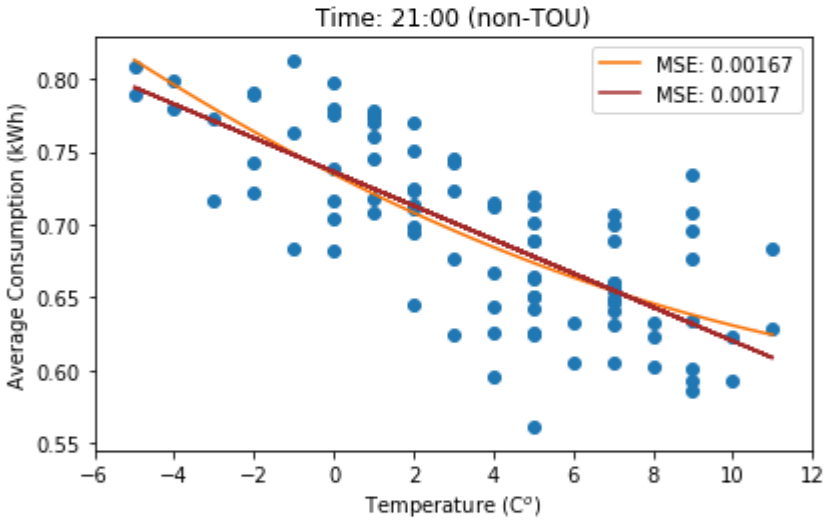

Time: 18:00 (TOU)
MSE: 0.00215
MSE: 0.0023
Average Consumption (kWh)
0.85
0.80
0.75
0.70
0.65
0.60
−2
0
2
4
6
8
10
12
Temperature (C°)

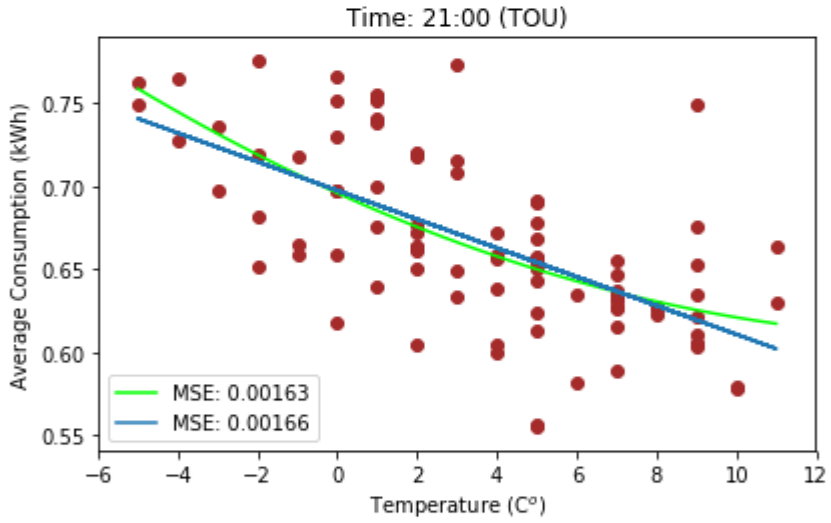

Time: 19:00 (non-TOU)
MSE: 0.0022
MSE: 0.00231
Average Consumption (kWh)
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
−4
−2
0
2
4
6
8
10
12
Temperature (C°)

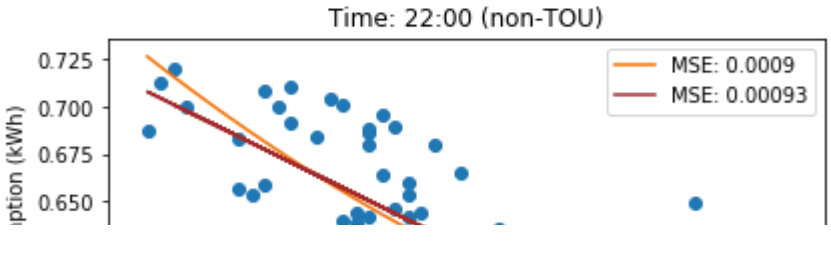

Time: 19:00 (TOU)
MSE: 0.00201
MSE: 0.00213
Average Consumption (kWh)
0.90
0.85
0.80
0.75
0.70
0.65
0.60
−4
−2
0
2
4
6
8
10
12
Temperature (C°)

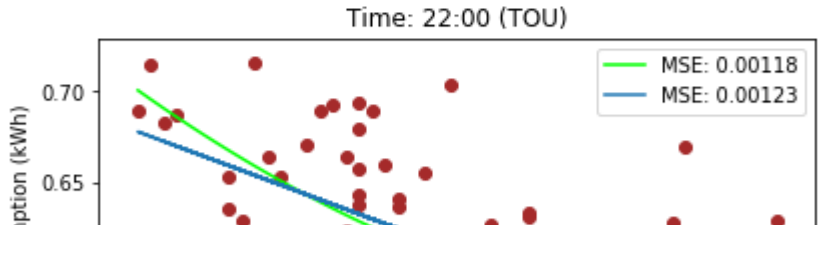

Time: 20:00 (non-TOU)
MSE: 0.00208
MSE: 0.00213
Average Consumption (kWh)
0.85
0.80
0.75
0.70
0.65
0.60
−4
−2
0
2
4
6
8
10
12
Temperature (C°)

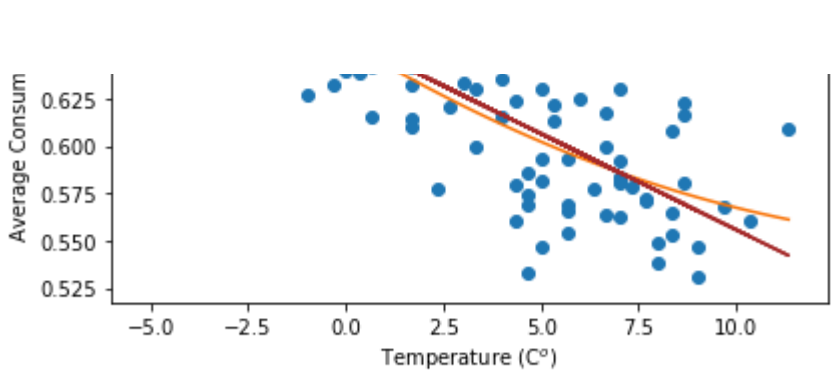

Time: 20:00 (TOU)
MSE: 0.00176
MSE: 0.0018
Average Consumption (kWh)
0.85
0.80
0.75
0.70
0.65
0.60
−4
−2
0
2
4
6
8
10
12
Temperature (C°)

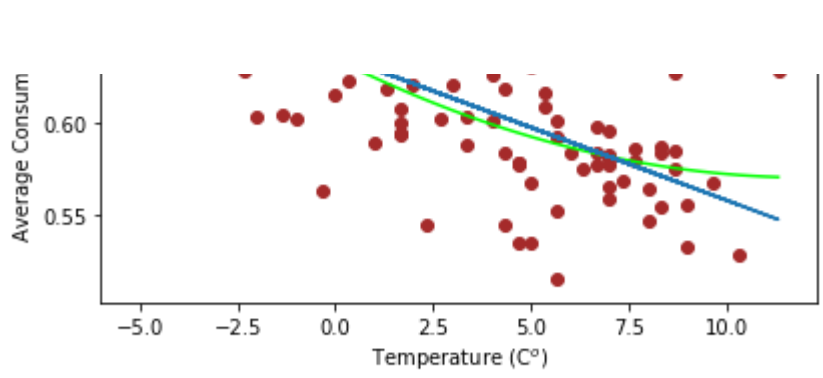

Time: 21:00 (non-TOU)
MSE: 0.00167
MSE: 0.0017
Average Consumption (kWh)
0.80
0.75
0.70
0.65
0.60
0.55
−6
−4
−2
0
2
4
6
8
10
12
Temperature (C°)

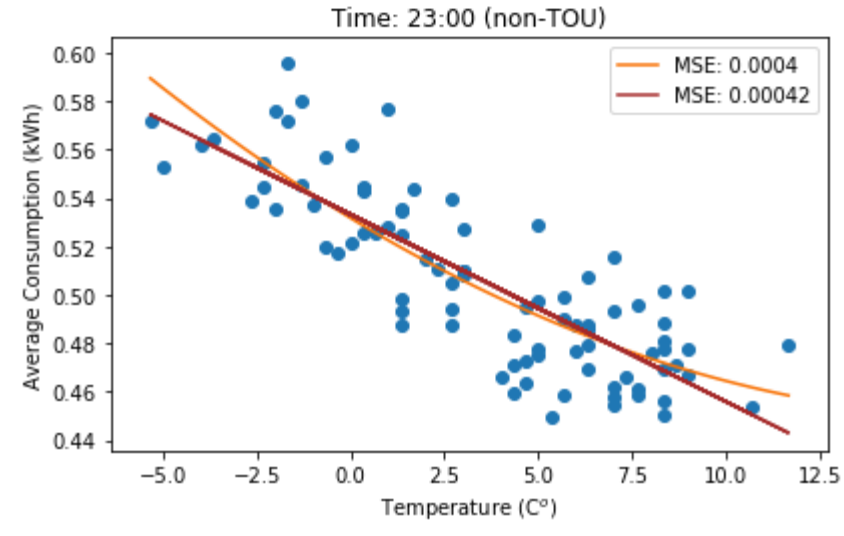

Time: 21:00 (TOU)
MSE: 0.00163
MSE: 0.00166
Average Consumption (kWh)
0.75
0.70
0.65
0.60
0.55
−6
−4
−2
0
2
4
6
8
10
12
Temperature (C°)

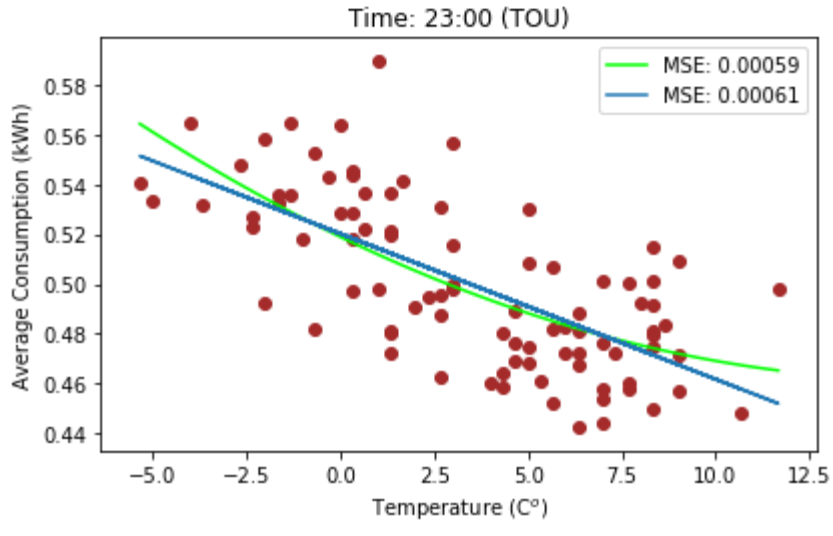

Time: 22:00 (non-TOU)
MSE: 0.0009
MSE: 0.00093
0.725
0.700
0.675
0.650


Time: 22:00 (TOU)
MSE: 0.00118
MSE: 0.00123
0.70
0.65

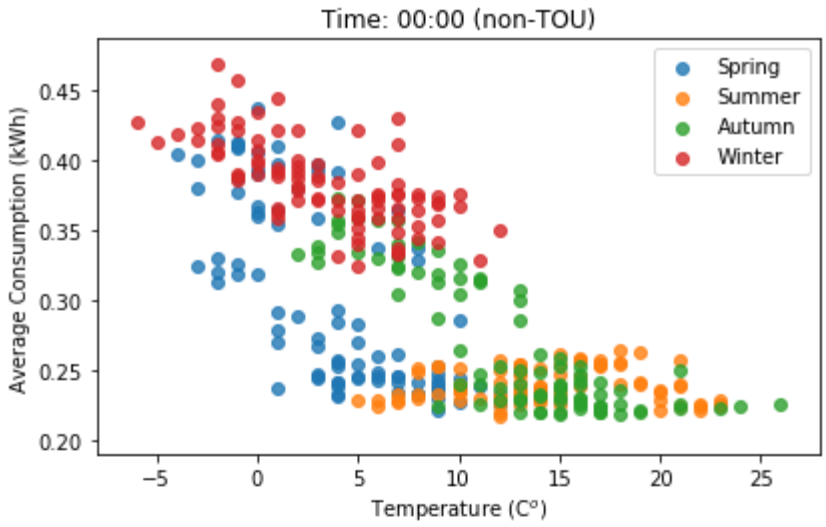

Average Consum
0.625
0.600
0.575
0.550
0.525
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
Temperature (C°)

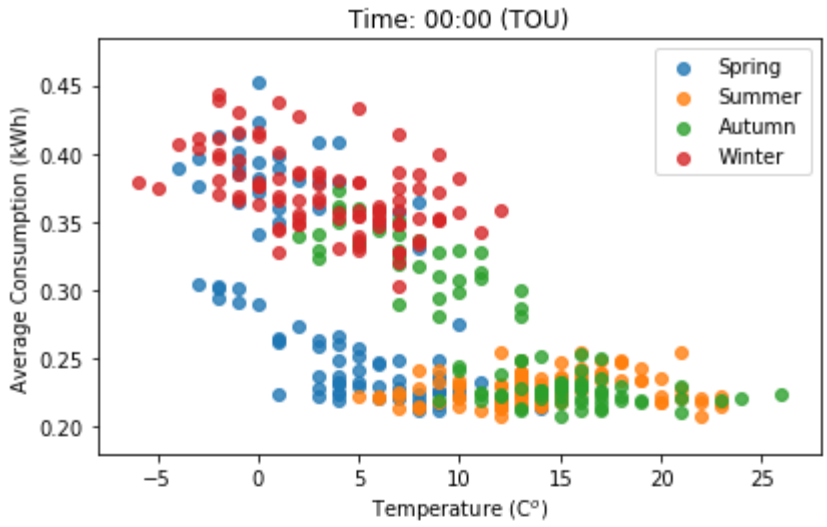

Average Consum
0.60
0.55
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
Temperature (C°)

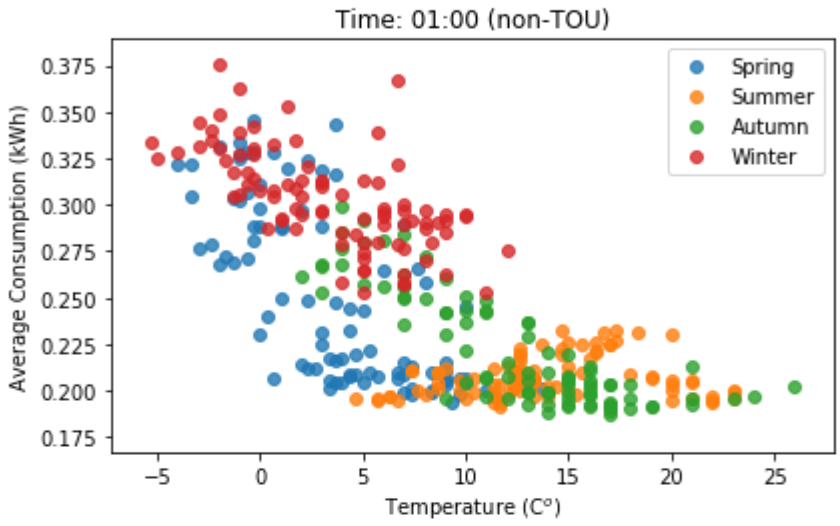

Time: 23:00 (non-TOU)
MSE: 0.0004
MSE: 0.00042
Average Consumption (kWh)
0.60
0.58
0.56
0.54
0.52
0.50
0.48
0.46
0.44
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
12.5
Temperature (C°)

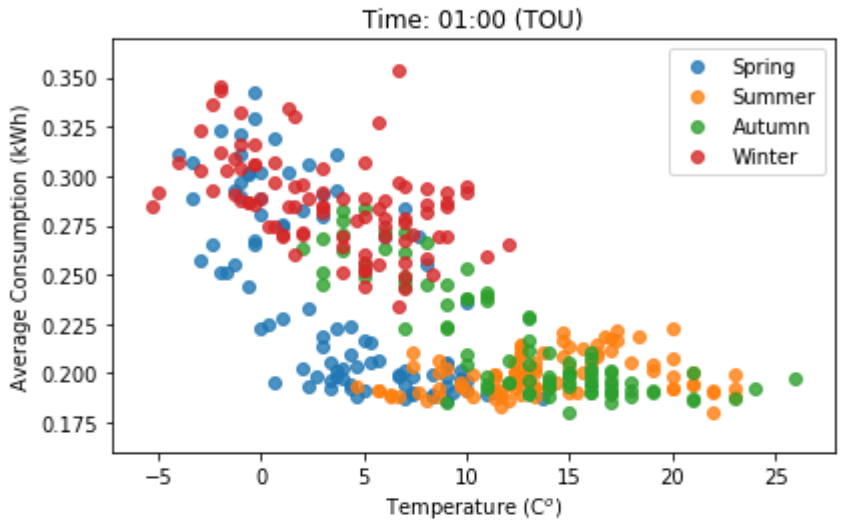

Time: 23:00 (TOU)
MSE: 0.00059
MSE: 0.00061
Average Consumption (kWh)
0.58
0.56
0.54
0.52
0.50
0.48
0.46
0.44
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
12.5
Temperature (C°)

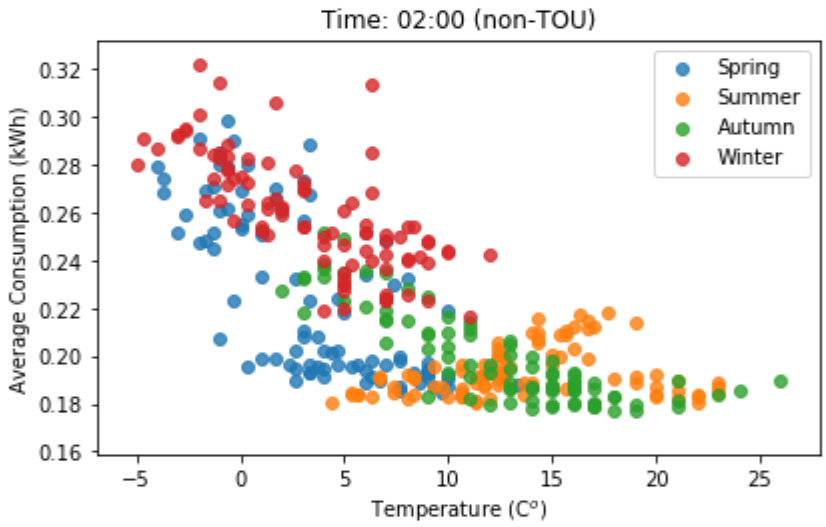
In [29]:
```
# Overview of the four seasons in the plot
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
# non-TOU vs TOU in terms of showing the elements in Average column such that they are hourly energy consumpt
ion v.s. temperature over 365 days
fig_all = plt.figure(figsize = (12,90))
ax_Ntou = [] # store subplot objects
ax_tou = [] # store subplot objects
lm = LinearRegression()
for i in range(24):
    ax_Ntou.append(fig_all.add_subplot(24, 2, 2 * i + 1))
    if i <= 9:
        x = df_wea1h_spring.TempC[df_wea1h_spring.GMT.str.contains('0' + str(i) + ':00:00')].values
        y = df_Ntou1h_spring.Average[df_Ntou1h_spring.GMT.str.contains('0' + str(i) + ':00:00')].values
        ax_Ntou[-1].scatter(x, y, label = 'Spring', alpha = 0.8)
        x = df_wea1h_summer.TempC[df_wea1h_summer.GMT.str.contains('0' + str(i) + ':00:00')].values
        y = df_Ntou1h_summer.Average[df_Ntou1h_summer.GMT.str.contains('0' + str(i) + ':00:00')].values
        ax_Ntou[-1].scatter(x, y, label = 'Summer', alpha = 0.8)
        x = df_wea1h_autumn.TempC[df_wea1h_autumn.GMT.str.contains('0' + str(i) + ':00:00')].values
        y = df_Ntou1h_autumn.Average[df_Ntou1h_autumn.GMT.str.contains('0' + str(i) + ':00:00')].values
        ax_Ntou[-1].scatter(x, y, label = 'Autumn', alpha = 0.8)
        x = df_wea1h_winter.TempC[df_wea1h_winter.GMT.str.contains('0' + str(i) + ':00:00')].values
        y = df_Ntou1h_winter.Average[df_Ntou1h_winter.GMT.str.contains('0' + str(i) + ':00:00')].values
        ax_Ntou[-1].scatter(x, y, label = 'Winter', alpha = 0.8)
        ax_Ntou[-1].set_title('Time: 0' + str(i) + ':00 (non-TOU)')
        ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_Ntou[-1].set_ylabel('Average Consumption (kWh)')
        ax_Ntou[-1].legend()
    else:
        x = df_wea1h_spring.TempC[df_wea1h_spring.GMT.str.contains(str(i) + ':00:00')].values
        y = df_Ntou1h_spring.Average[df_Ntou1h_spring.GMT.str.contains(str(i) + ':00:00')].values
        ax_Ntou[-1].scatter(x, y, label = 'Spring', alpha = 0.8)
        x = df_wea1h_summer.TempC[df_wea1h_summer.GMT.str.contains(str(i) + ':00:00')].values
        y = df_Ntou1h_summer.Average[df_Ntou1h_summer.GMT.str.contains(str(i) + ':00:00')].values
        ax_Ntou[-1].scatter(x, y, label = 'Summer', alpha = 0.8)
        x = df_wea1h_autumn.TempC[df_wea1h_autumn.GMT.str.contains(str(i) + ':00:00')].values
        y = df_Ntou1h_autumn.Average[df_Ntou1h_autumn.GMT.str.contains(str(i) + ':00:00')].values
        ax_Ntou[-1].scatter(x, y, label = 'Autumn', alpha = 0.8)
        x = df_wea1h_winter.TempC[df_wea1h_winter.GMT.str.contains(str(i) + ':00:00')].values
        y = df_Ntou1h_winter.Average[df_Ntou1h_winter.GMT.str.contains(str(i) + ':00:00')].values
        ax_Ntou[-1].scatter(x, y, label = 'Winter', alpha = 0.8)
        ax_Ntou[-1].set_title('Time: ' + str(i) + ':00 (non-TOU)')
        ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_Ntou[-1].set_ylabel('Average Consumption (kWh)')
        ax_Ntou[-1].legend()
for i in range(24):
    ax_tou.append(fig_all.add_subplot(24, 2, 2 * i + 2))
    if i <= 9:
        x = df_wea1h_spring.TempC[df_wea1h_spring.GMT.str.contains('0' + str(i) + ':00:00')].values
        y = df_tou1h_spring.Average[df_tou1h_spring.GMT.str.contains('0' + str(i) + ':00:00')].values
        ax_tou[-1].scatter(x, y, label = 'Spring', alpha = 0.8)
        x = df_wea1h_summer.TempC[df_wea1h_summer.GMT.str.contains('0' + str(i) + ':00:00')].values
        y = df_tou1h_summer.Average[df_tou1h_summer.GMT.str.contains('0' + str(i) + ':00:00')].values
        ax_tou[-1].scatter(x, y, label = 'Summer', alpha = 0.8)
        x = df_wea1h_autumn.TempC[df_wea1h_autumn.GMT.str.contains('0' + str(i) + ':00:00')].values
        y = df_tou1h_autumn.Average[df_tou1h_autumn.GMT.str.contains('0' + str(i) + ':00:00')].values
        ax_tou[-1].scatter(x, y, label = 'Autumn', alpha = 0.8)
        x = df_wea1h_winter.TempC[df_wea1h_winter.GMT.str.contains('0' + str(i) + ':00:00')].values
        y = df_tou1h_winter.Average[df_tou1h_winter.GMT.str.contains('0' + str(i) + ':00:00')].values
        ax_tou[-1].scatter(x, y, label = 'Winter', alpha = 0.8)
        ax_tou[-1].set_title('Time: 0' + str(i) + ':00 (TOU)')
        ax_tou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_tou[-1].set_ylabel('Average Consumption (kWh)')
        ax_tou[-1].legend()
    else:
        x = df_wea1h_spring.TempC[df_wea1h_spring.GMT.str.contains(str(i) + ':00:00')].values
        y = df_tou1h_spring.Average[df_tou1h_spring.GMT.str.contains(str(i) + ':00:00')].values
        ax_tou[-1].scatter(x, y, label = 'Spring')
        x = df_wea1h_summer.TempC[df_wea1h_summer.GMT.str.contains(str(i) + ':00:00')].values
        y = df_tou1h_summer.Average[df_tou1h_summer.GMT.str.contains(str(i) + ':00:00')].values
        ax_tou[-1].scatter(x, y, label = 'Summer')
        x = df_wea1h_autumn.TempC[df_wea1h_autumn.GMT.str.contains(str(i) + ':00:00')].values
        y = df_tou1h_autumn.Average[df_tou1h_autumn.GMT.str.contains(str(i) + ':00:00')].values
        ax_tou[-1].scatter(x, y, label = 'Autumn')
        x = df_wea1h_winter.TempC[df_wea1h_winter.GMT.str.contains(str(i) + ':00:00')].values
        y = df_tou1h_winter.Average[df_tou1h_winter.GMT.str.contains(str(i) + ':00:00')].values
        ax_tou[-1].scatter(x, y, label = 'Winter')
        ax_tou[-1].set_title('Time: ' + str(i) + ':00 (TOU)')
        ax_tou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_tou[-1].set_ylabel('Average Consumption (kWh)')
        ax_tou[-1].legend()
plt.tight_layout()
```

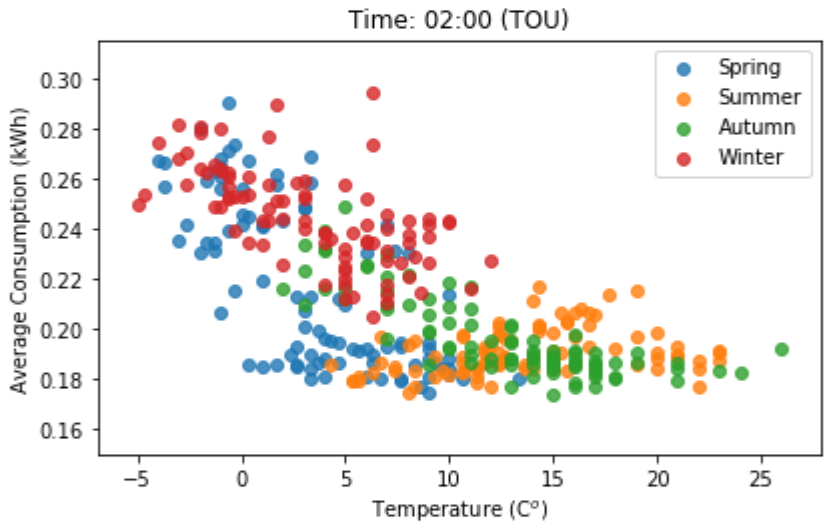

Time: 00:00 (non-TOU)
Average Consumption (kWh)
Temperature (C°)
Spring
Summer
Autumn
Winter

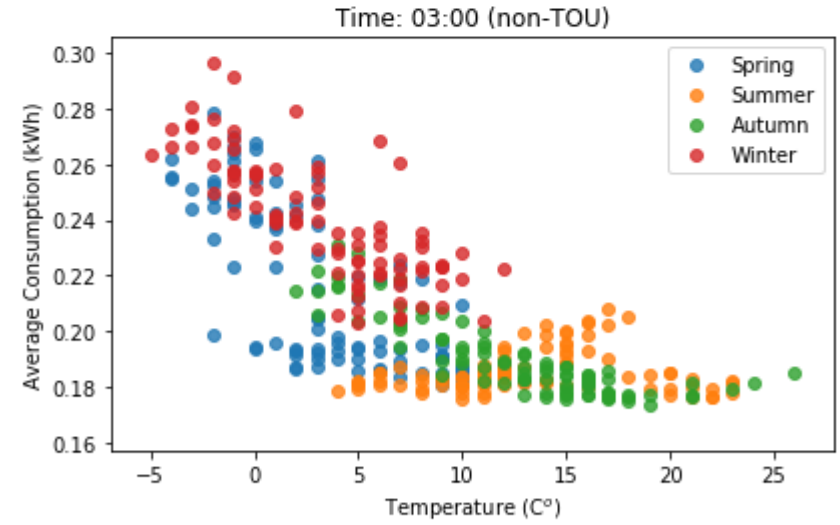

Time: 00:00 (TOU)
Average Consumption (kWh)
Temperature (C°)
Spring
Summer
Autumn
Winter

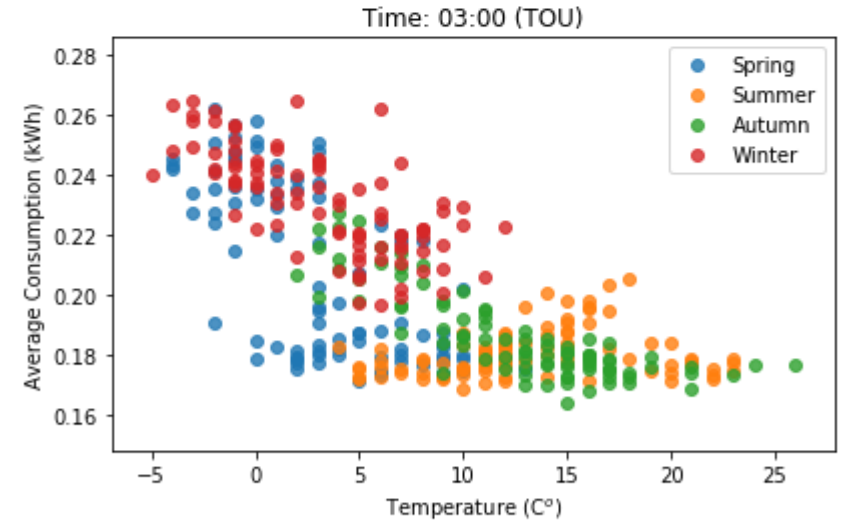

Time: 01:00 (non-TOU)
Average Consumption (kWh)
Temperature (C°)
Spring
Summer
Autumn
Winter

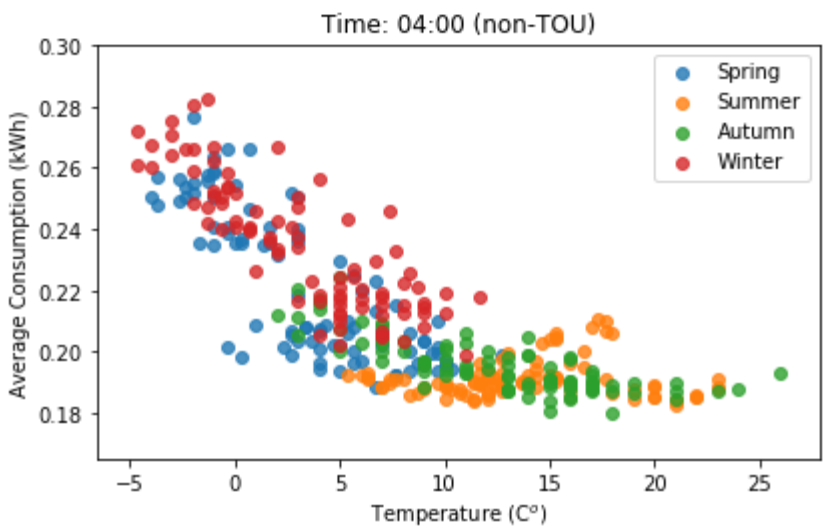

Time: 01:00 (TOU)
Average Consumption (kWh)
Temperature (C°)
Spring
Summer
Autumn
Winter

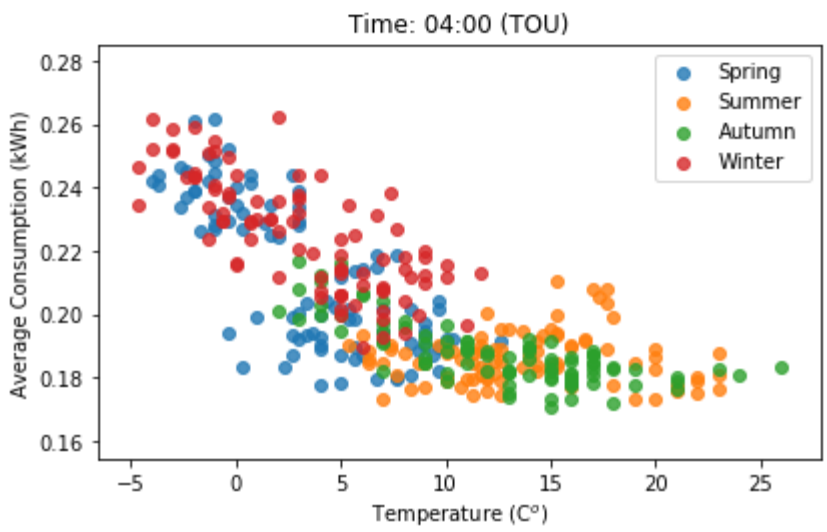

Time: 02:00 (non-TOU)
Average Consumption (kWh)
Temperature (C°)
Spring
Summer
Autumn
Winter

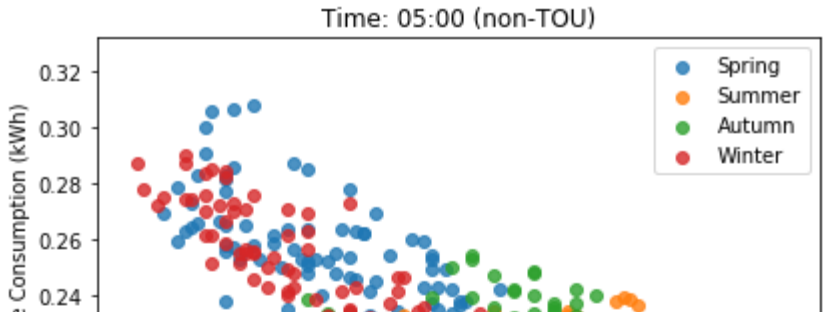

Time: 02:00 (TOU)
Average Consumption (kWh)
Temperature (C°)
Spring
Summer
Autumn
Winter

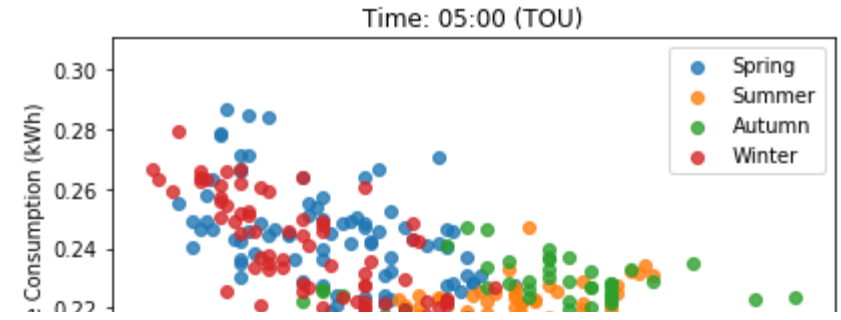

Time: 03:00 (non-TOU)
Average Consumption (kWh)
Temperature (C°)
Spring
Summer
Autumn
Winter

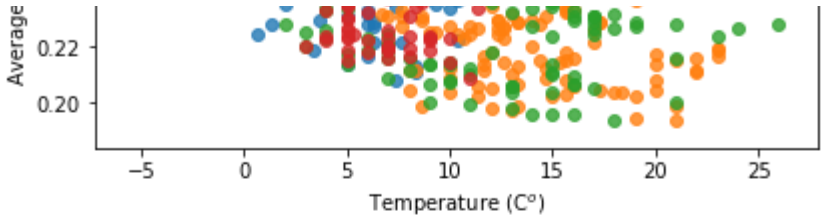

Time: 03:00 (TOU)
Average Consumption (kWh)
Temperature (C°)
Spring
Summer
Autumn
Winter

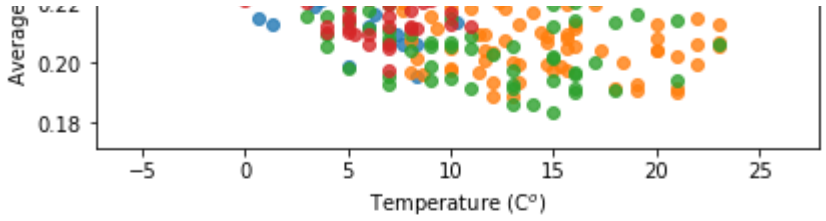

Time: 04:00 (non-TOU)
Average Consumption (kWh)
Temperature (C°)
Spring
Summer
Autumn
Winter

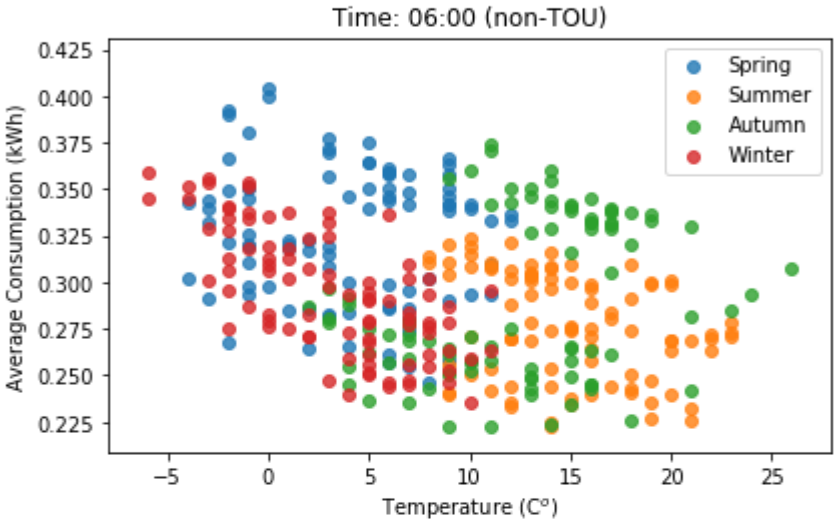

Time: 04:00 (TOU)
Average Consumption (kWh)
Temperature (C°)
Spring
Summer
Autumn
Winter

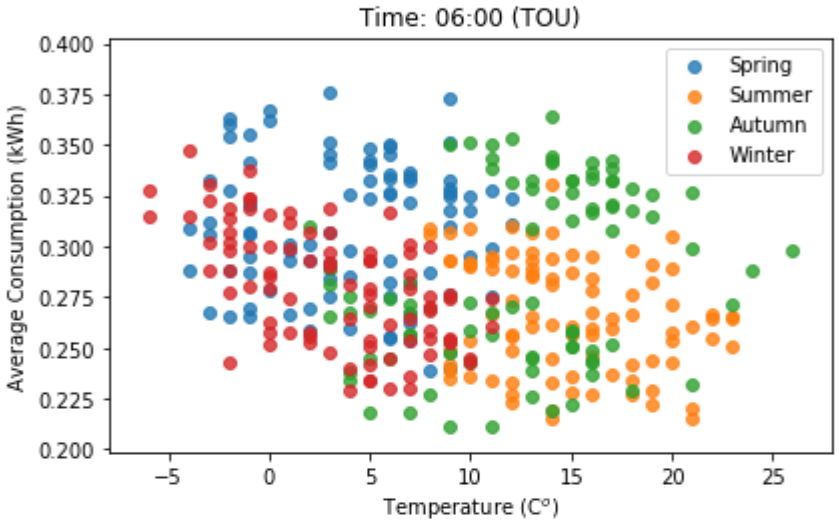

Time: 05:00 (non-TOU)
Spring
Summer
Autumn
Winter

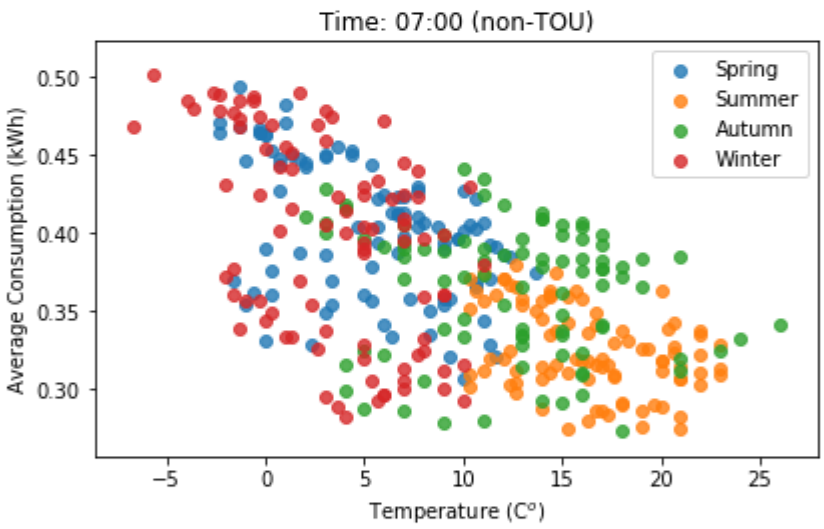

Time: 05:00 (TOU)
Spring
Summer
Autumn
Winter

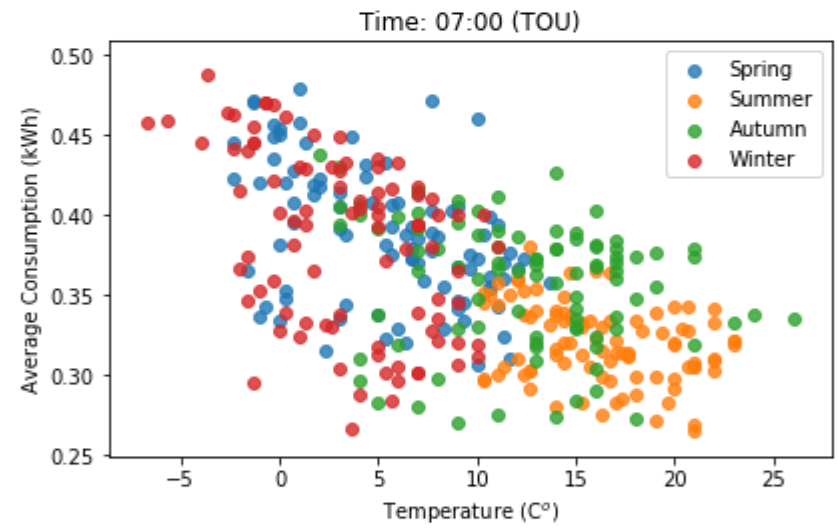

Average
0.22
0.20
−5
0
5
10
15
20
25
Temperature (C°)

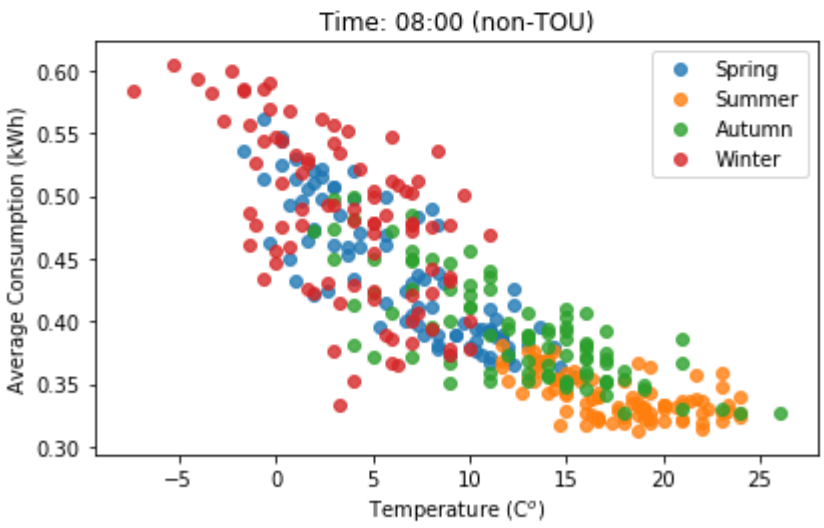

Average
0.20
0.18
−5
0
5
10
15
20
25
Temperature (C°)

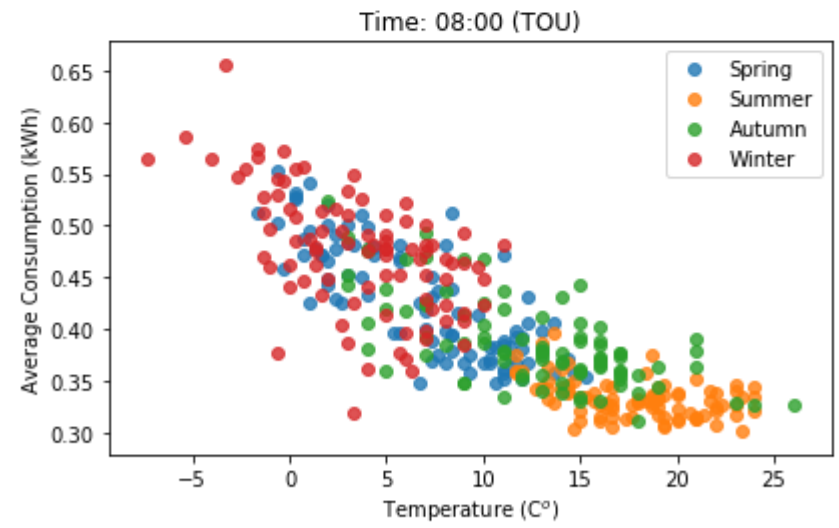

Time: 06:00 (non-TOU)
Average Consumption (kWh)
0.425
0.400
0.375
0.350
0.325
0.300
0.275
0.250
0.225
Spring
Summer
Autumn
Winter
−5
0
5
10
15
20
25
Temperature (C°)

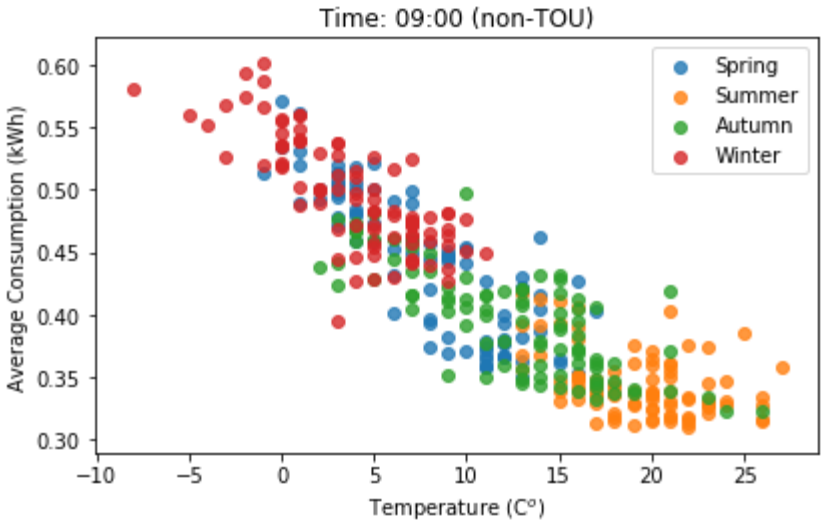

Time: 06:00 (TOU)
Average Consumption (kWh)
0.400
0.375
0.350
0.325
0.300
0.275
0.250
0.225
0.200
Spring
Summer
Autumn
Winter
−5
0
5
10
15
20
25
Temperature (C°)

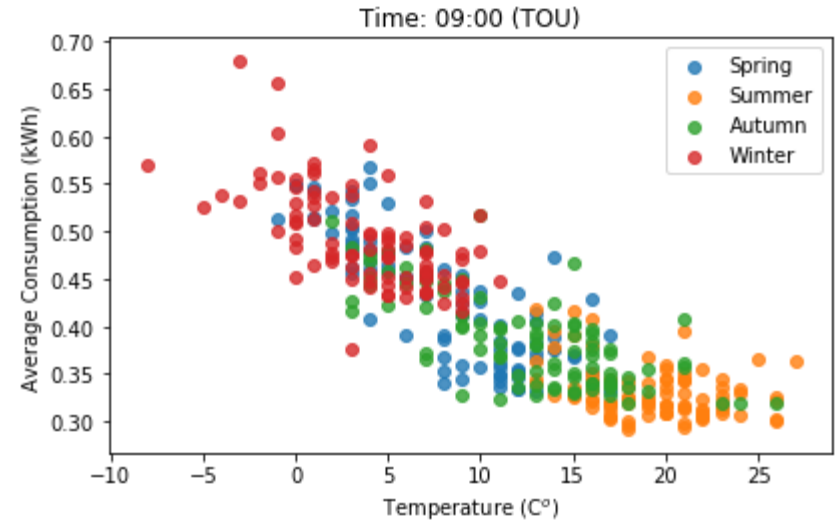

Time: 07:00 (non-TOU)
Average Consumption (kWh)
0.50
0.45
0.40
0.35
0.30
Spring
Summer
Autumn
Winter
−5
0
5
10
15
20
25
Temperature (C°)

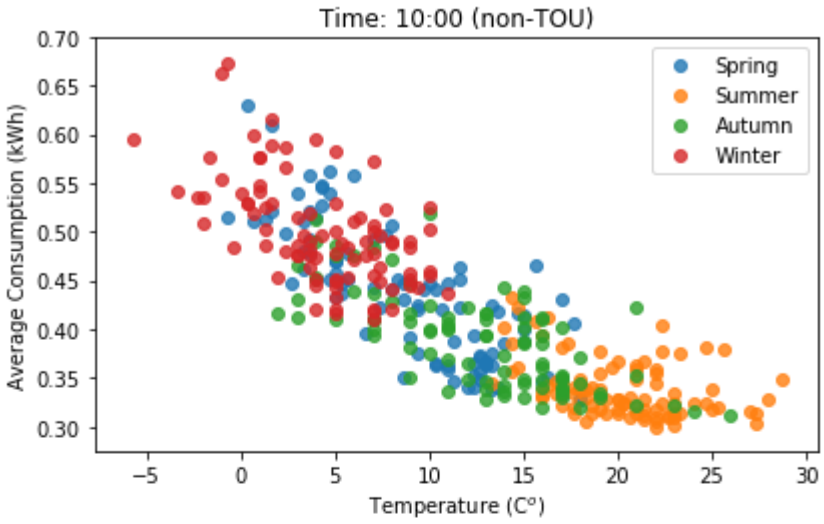

Time: 07:00 (TOU)
Average Consumption (kWh)
0.50
0.45
0.40
0.35
0.30
0.25
Spring
Summer
Autumn
Winter
−5
0
5
10
15
20
25
Temperature (C°)

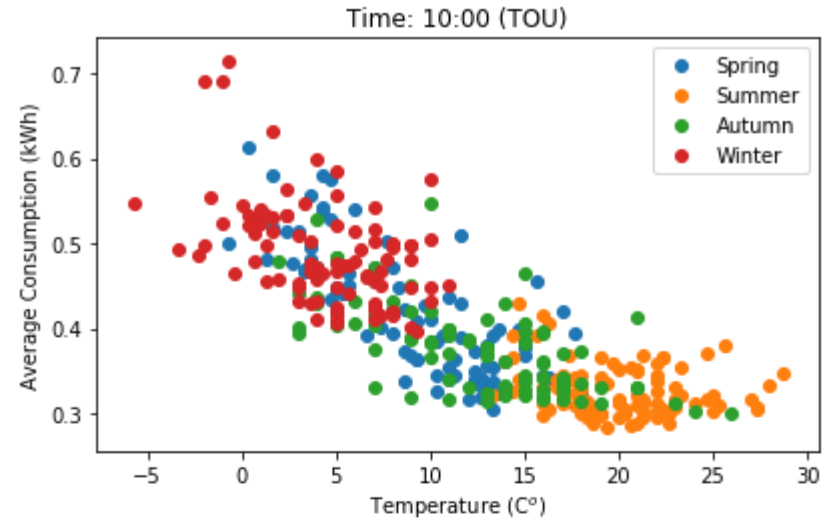

Time: 08:00 (non-TOU)
Average Consumption (kWh)
0.60
0.55
0.50
0.45
0.40
0.35
0.30
Spring
Summer
Autumn
Winter
−5
0
5
10
15
20
25
Temperature (C°)

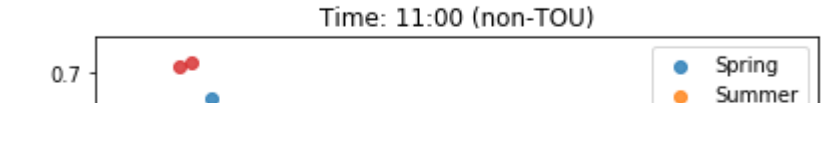

Time: 08:00 (TOU)
Average Consumption (kWh)
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
Spring
Summer
Autumn
Winter
−5
0
5
10
15
20
25
Temperature (C°)

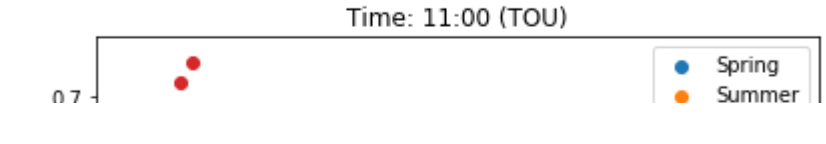

Time: 09:00 (non-TOU)
Average Consumption (kWh)
0.60
0.55
0.50
0.45
0.40
0.35
0.30
Spring
Summer
Autumn
Winter
−10
−5
0
5
10
15
20
25
Temperature (C°)

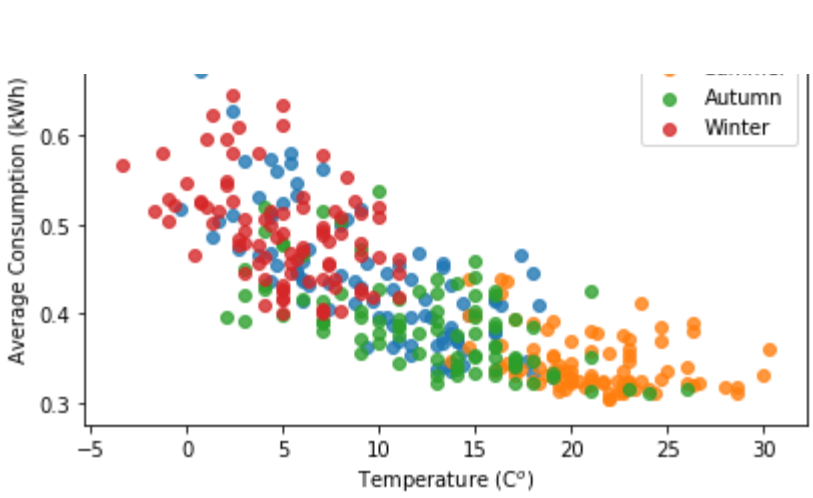

Time: 09:00 (TOU)
Average Consumption (kWh)
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
Spring
Summer
Autumn
Winter
−10
−5
0
5
10
15
20
25
Temperature (C°)

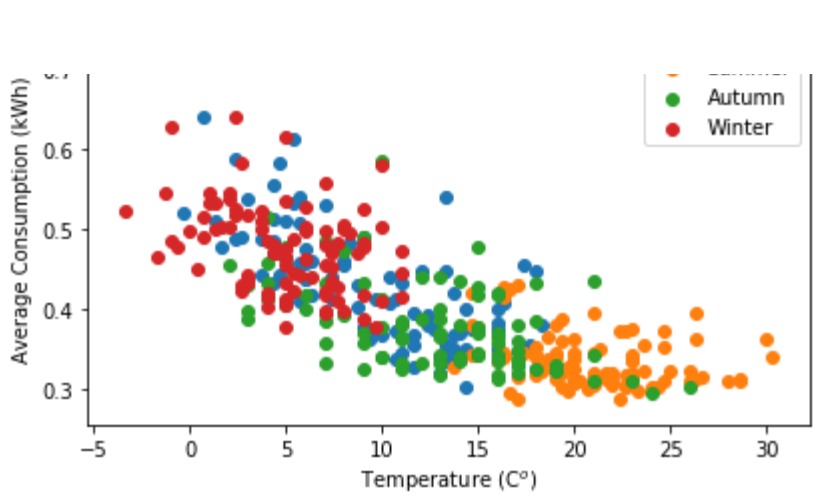

Time: 10:00 (non-TOU)
Average Consumption (kWh)
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
Spring
Summer
Autumn
Winter
−5
0
5
10
15
20
25
30
Temperature (C°)

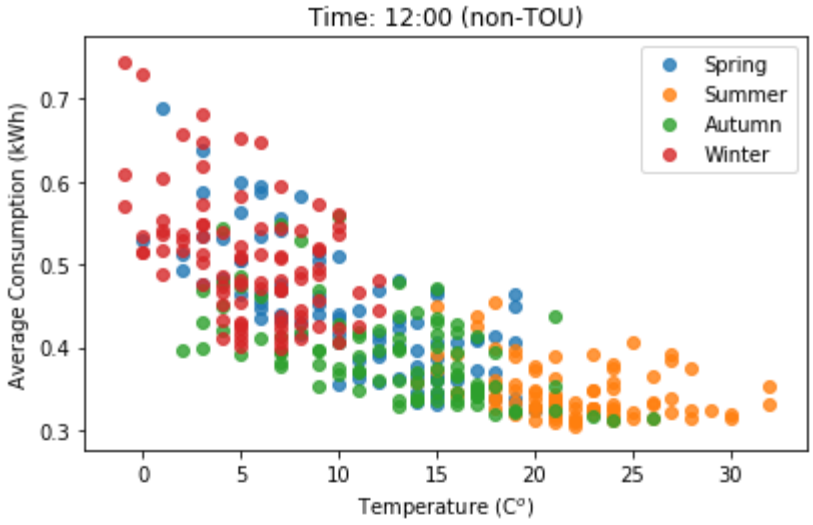

Time: 10:00 (TOU)
Average Consumption (kWh)
0.7
0.6
0.5
0.4
0.3
Spring
Summer
Autumn
Winter
−5
0
5
10
15
20
25
30
Temperature (C°)

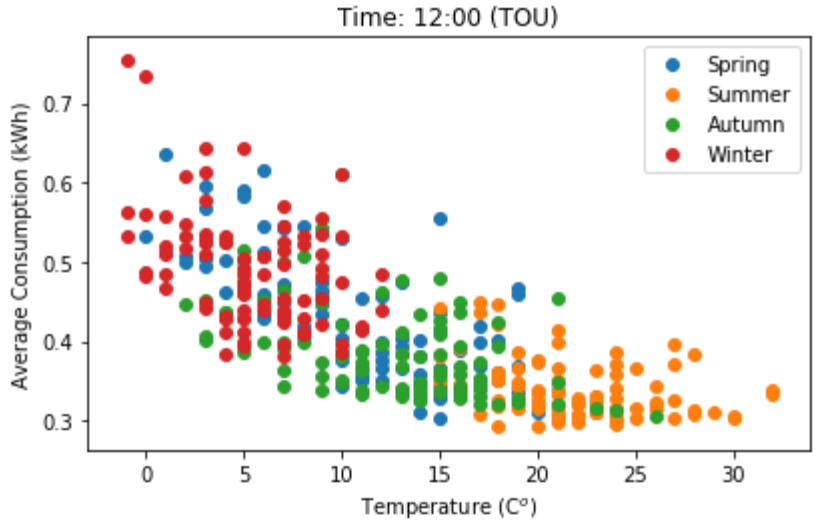

Time: 11:00 (non-TOU)
0.7
Spring
Summer

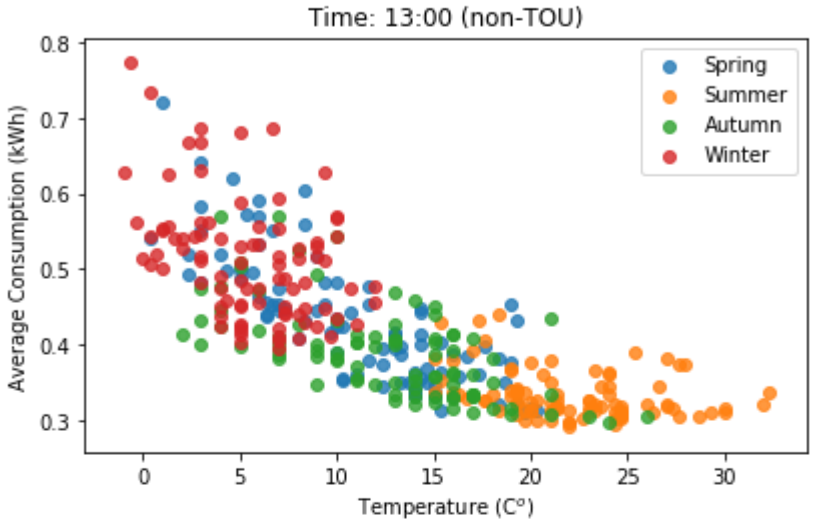

Time: 11:00 (TOU)
0.7
Spring
Summer

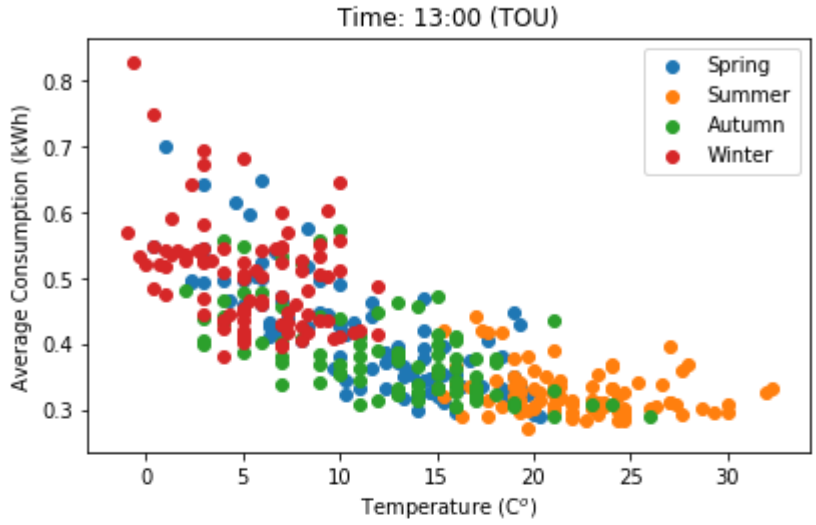

Autumn
Winter
Average Consumption (kWh)
Temperature (C°)

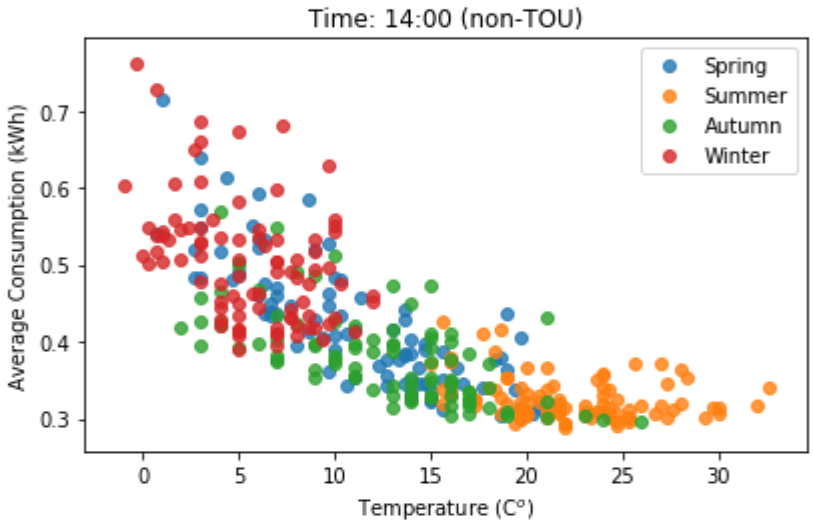

Autumn
Winter
Average Consumption (kWh)
Temperature (C°)

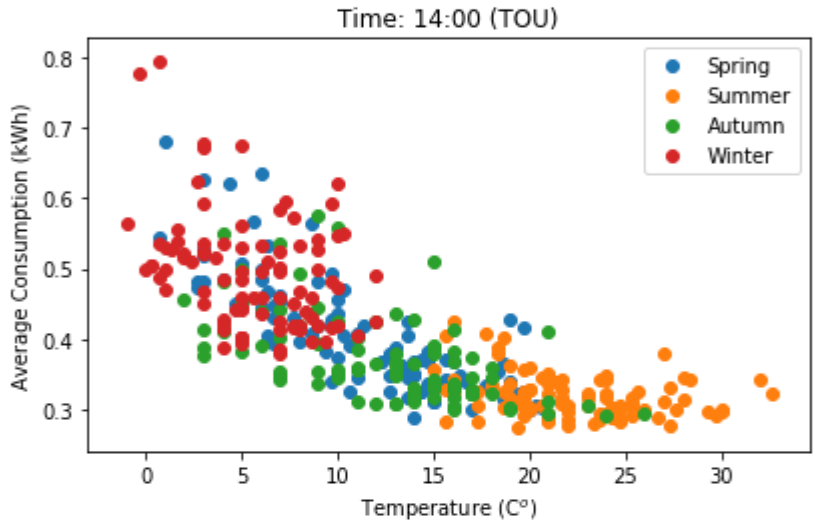

Time: 12:00 (non-TOU)
Spring
Summer
Autumn
Winter
Average Consumption (kWh)
Temperature (C°)

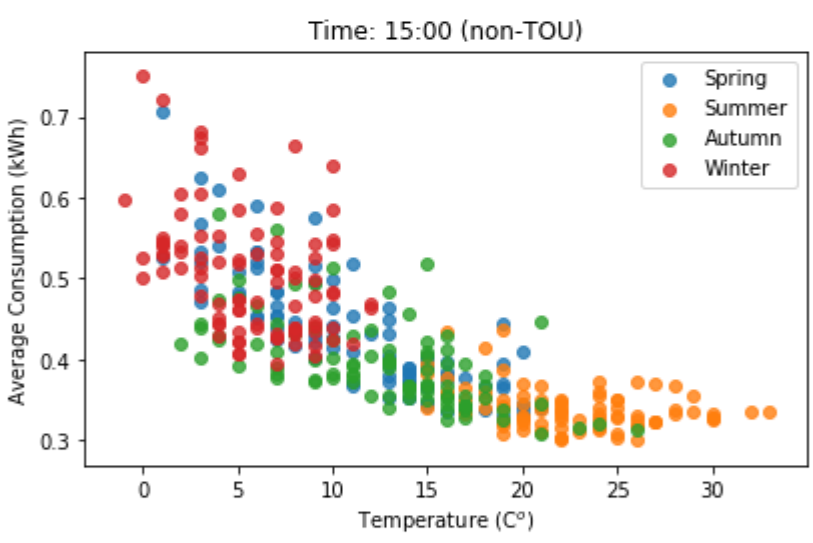

Time: 12:00 (TOU)
Spring
Summer
Autumn
Winter
Average Consumption (kWh)
Temperature (C°)

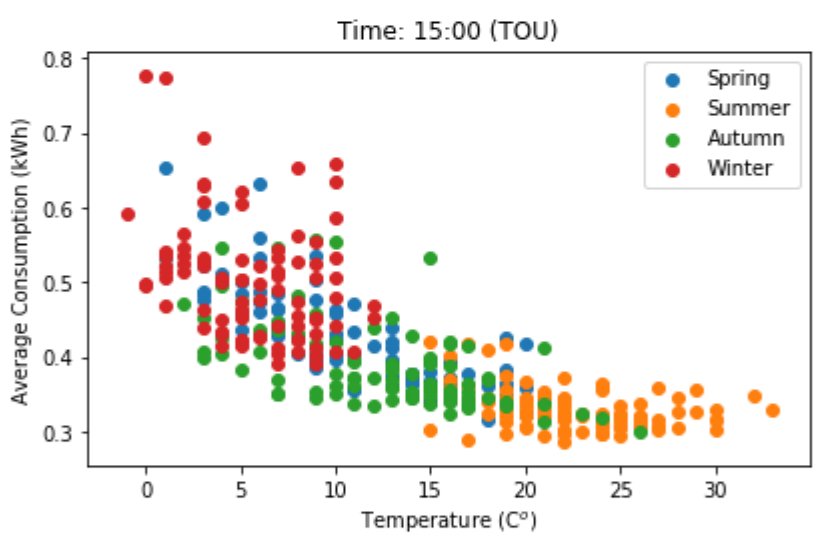

Time: 13:00 (non-TOU)
Spring
Summer
Autumn
Winter
Average Consumption (kWh)
Temperature (C°)

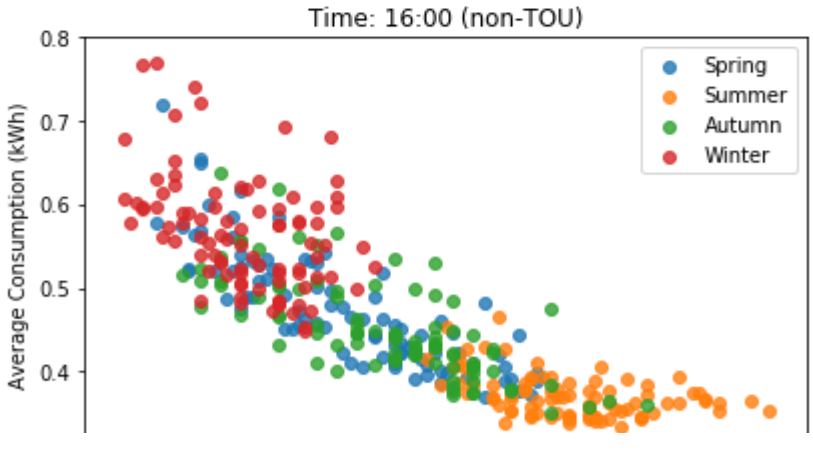

Time: 13:00 (TOU)
Spring
Summer
Autumn
Winter
Average Consumption (kWh)
Temperature (C°)

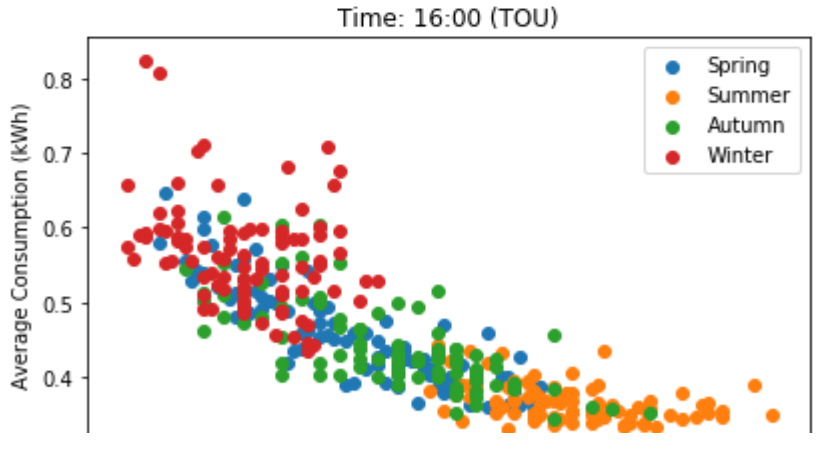

Time: 14:00 (non-TOU)
Spring
Summer
Autumn
Winter
Average Consumption (kWh)
Temperature (C°)

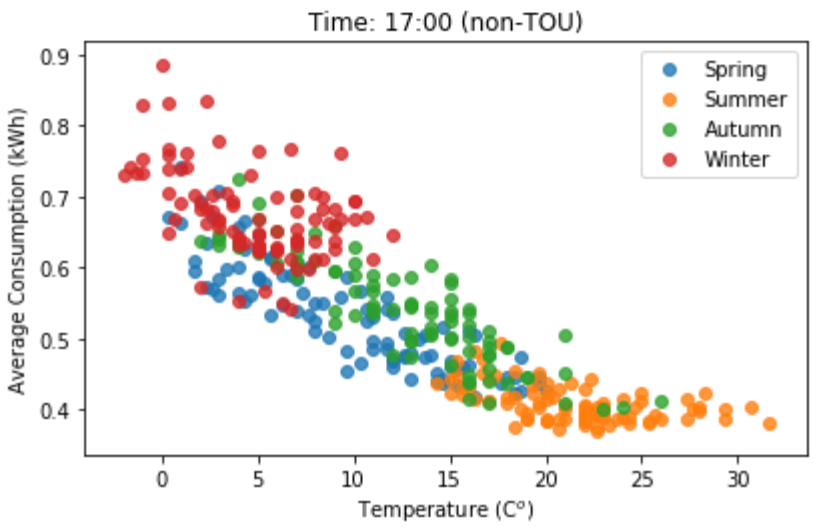

Time: 14:00 (TOU)
Spring
Summer
Autumn
Winter
Average Consumption (kWh)
Temperature (C°)

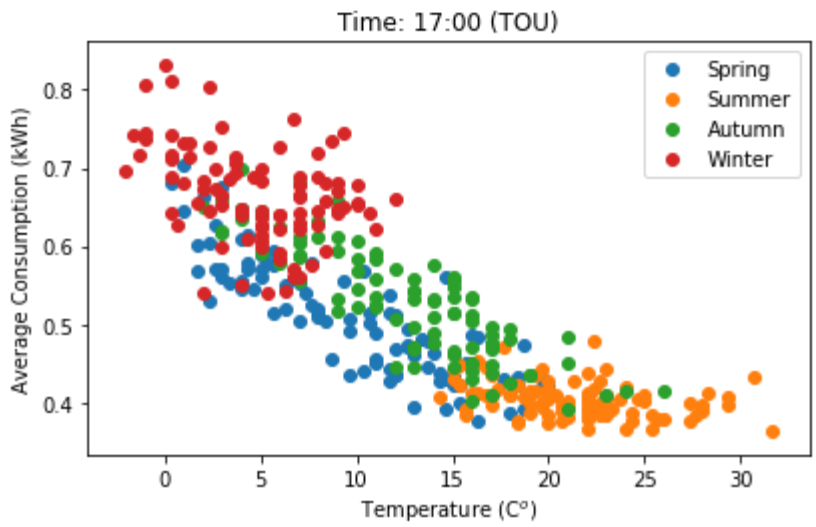

Time: 15:00 (non-TOU)
Spring
Summer
Autumn
Winter
Average Consumption (kWh)
Temperature (C°)

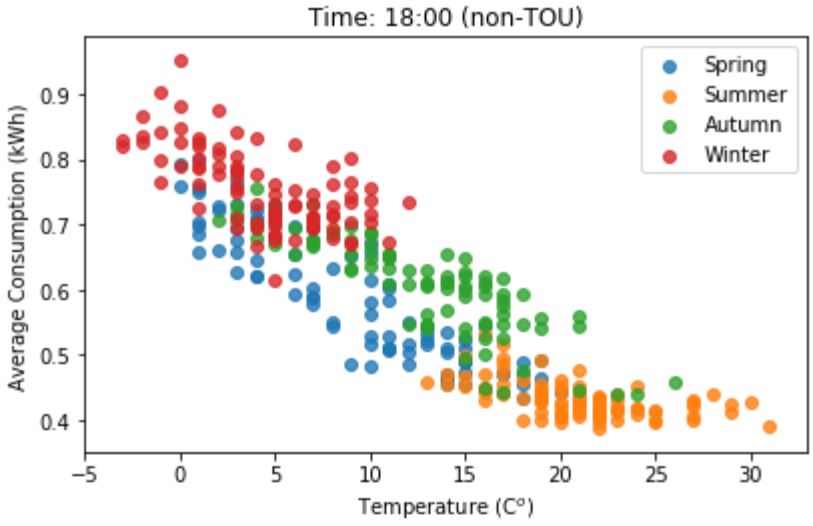

Time: 15:00 (TOU)
Spring
Summer
Autumn
Winter
Average Consumption (kWh)
Temperature (C°)

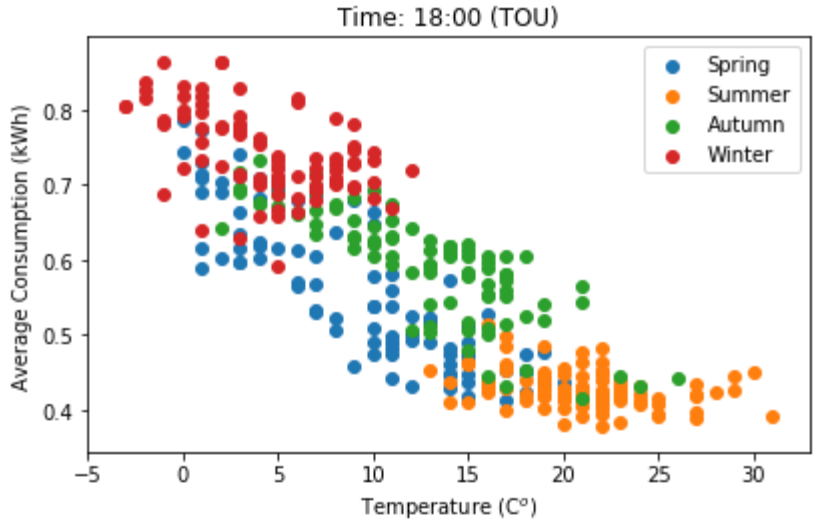

Time: 16:00 (non-TOU)
Spring
Summer
Autumn
Winter
Average Consumption (kWh)

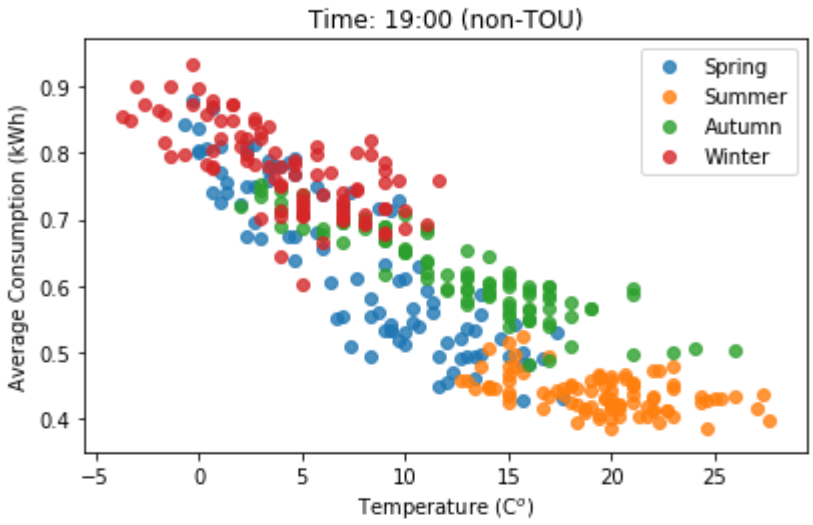

Time: 16:00 (TOU)
Spring
Summer
Autumn
Winter
Average Consumption (kWh)

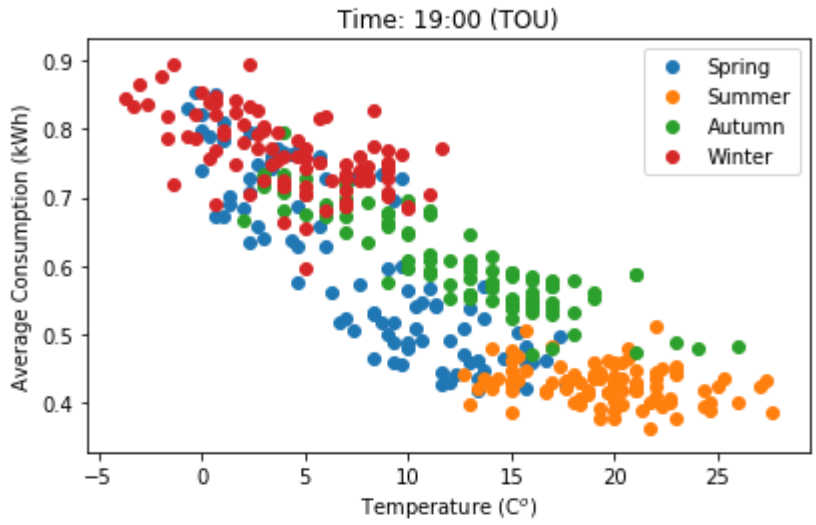

Time: 17:00 (non-TOU)
Average Consumption (kWh)
Temperature (C°)
Spring
Summer
Autumn
Winter

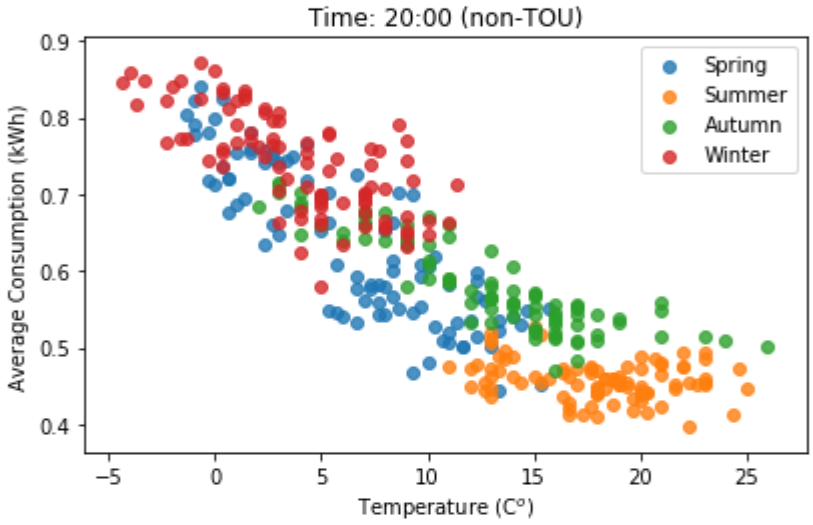

Time: 17:00 (TOU)
Average Consumption (kWh)
Temperature (C°)
Spring
Summer
Autumn
Winter

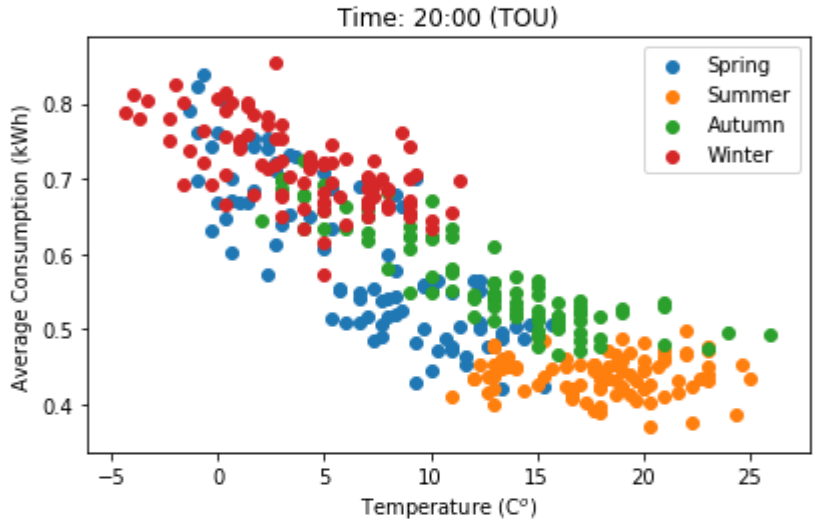

Time: 18:00 (non-TOU)
Average Consumption (kWh)
Temperature (C°)
Spring
Summer
Autumn
Winter

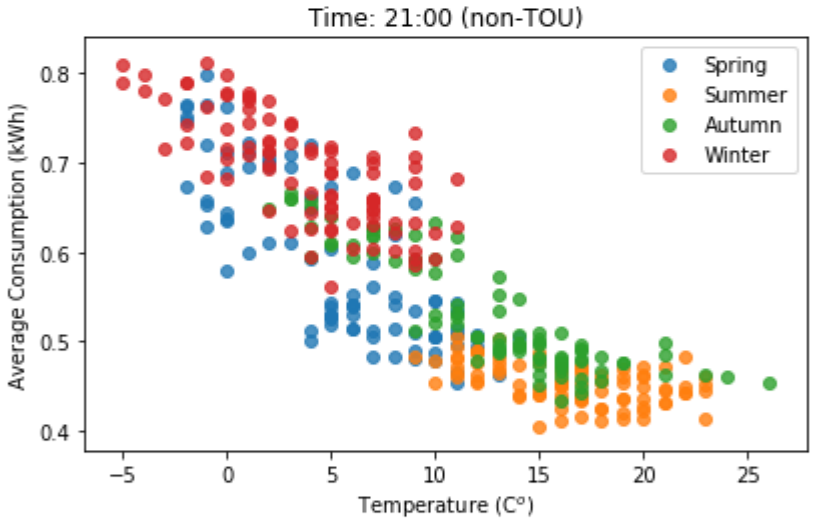

Time: 18:00 (TOU)
Average Consumption (kWh)
Temperature (C°)
Spring
Summer
Autumn
Winter

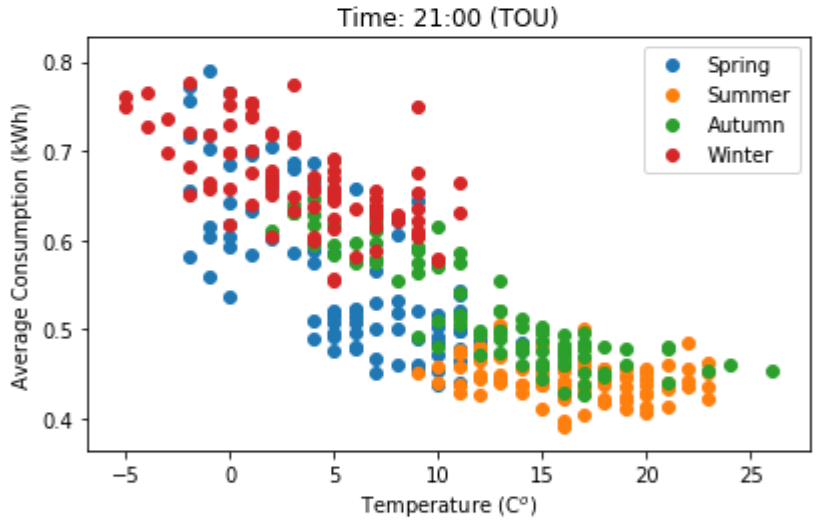

Time: 19:00 (non-TOU)
Average Consumption (kWh)
Temperature (C°)
Spring
Summer
Autumn
Winter


Time: 19:00 (TOU)
Average Consumption (kWh)
Temperature (C°)
Spring
Summer
Autumn
Winter


Time: 20:00 (non-TOU)
Average Consumption (kWh)
Temperature (C°)
Spring
Summer
Autumn
Winter

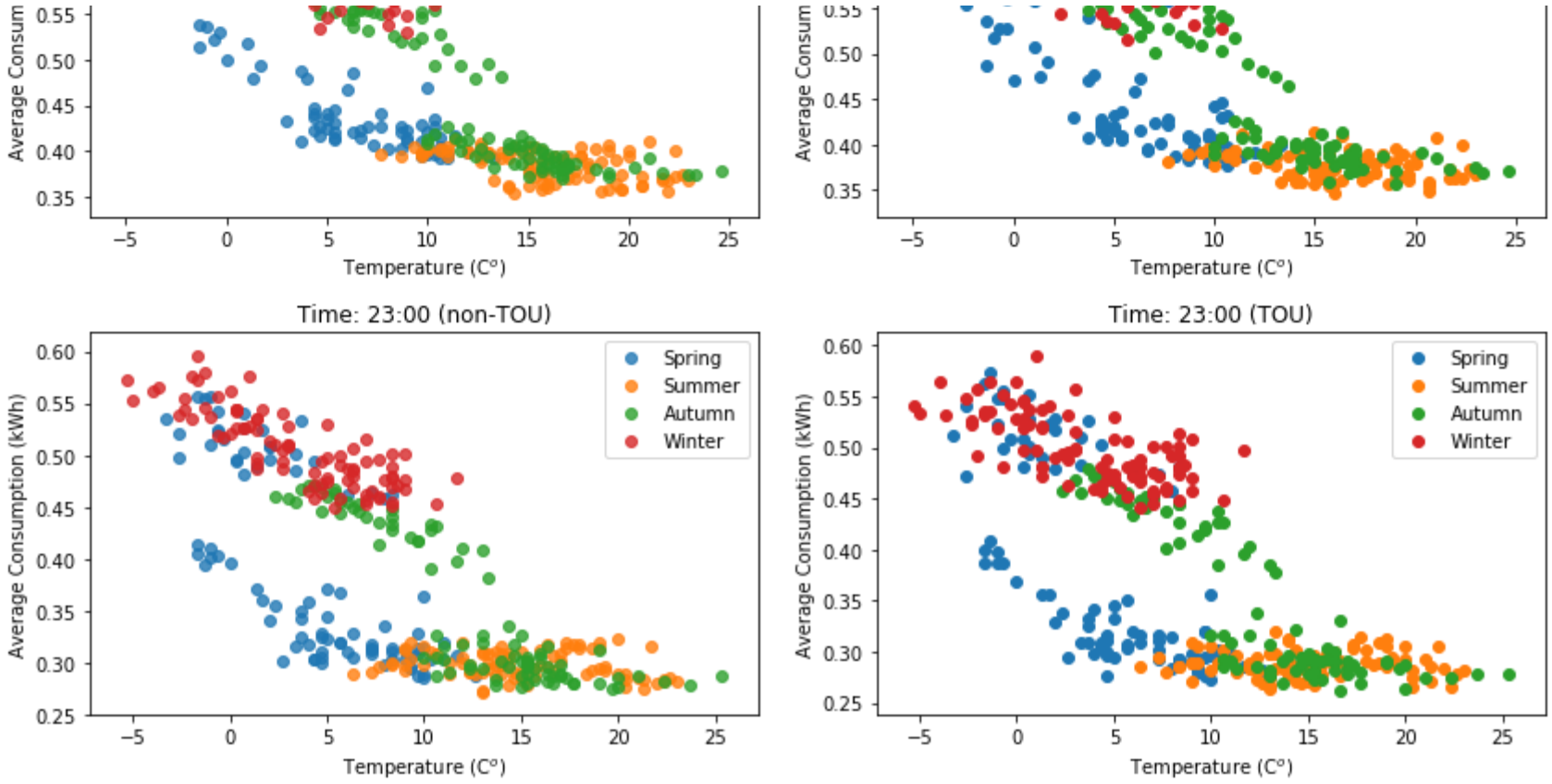

Time: 20:00 (TOU)
Average Consumption (kWh)
Temperature (C°)
Spring
Summer
Autumn
Winter


Time: 21:00 (non-TOU)
Average Consumption (kWh)
Temperature (C°)
Spring
Summer
Autumn
Winter


Time: 21:00 (TOU)
Average Consumption (kWh)
Temperature (C°)
Spring
Summer
Autumn
Winter

Time: 22:00 (non-TOU)
Temperature (C°)
Spring
Summer
Autumn
Winter

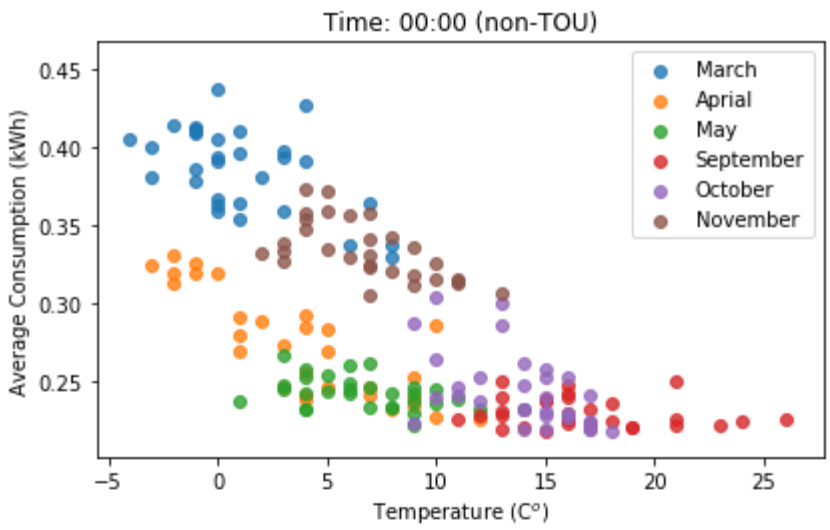
Time: 22:00 (TOU)
Temperature (C°)
Spring
Summer
Autumn
Winter

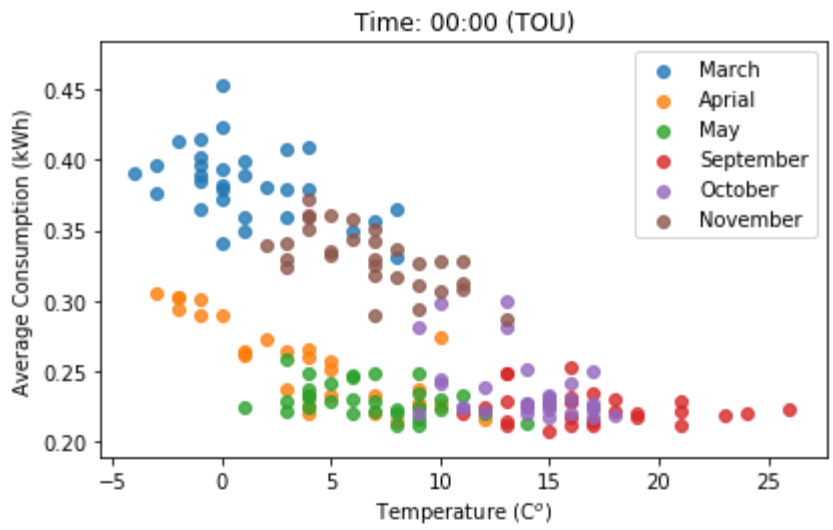


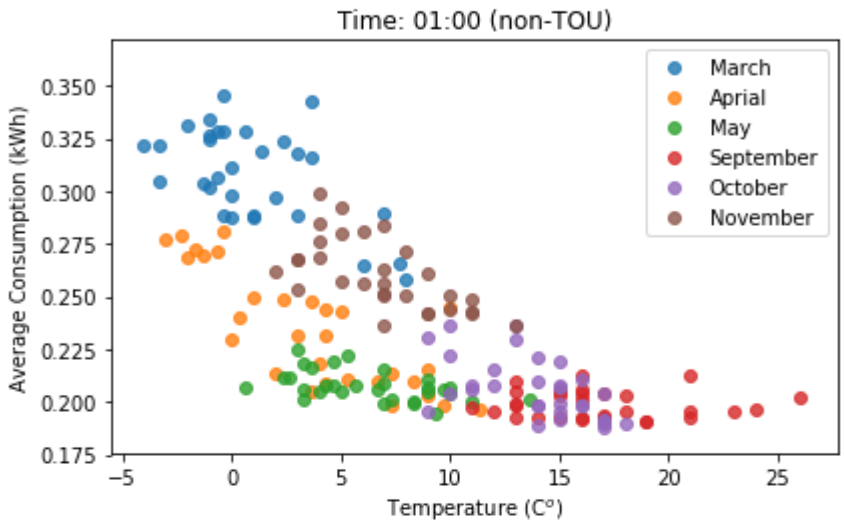
We can see from the above pictures that summer and winter are usually seperated points, so we can focus on the other two seasons, Spring and Autumn, and divide the data by different months (a finer partition)

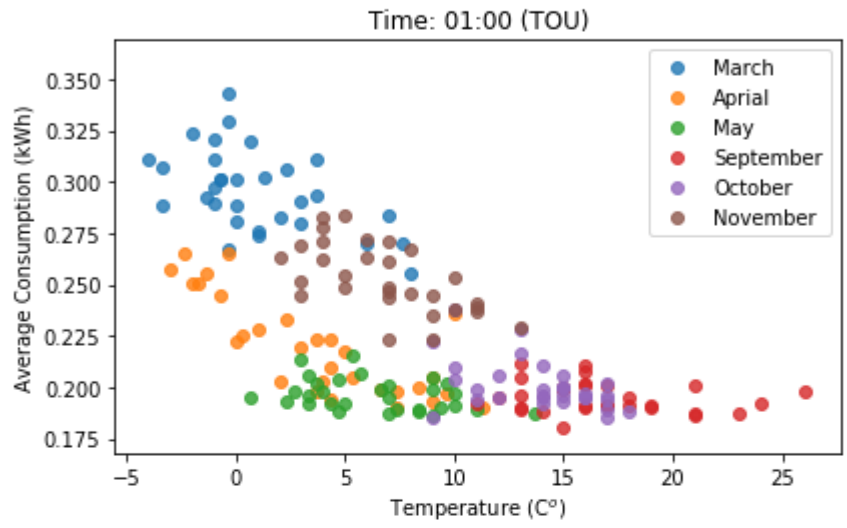
In [32]:
```
# Create Spring, Summer, Autumn and Winter dataframe for load, price, weather
spring = [3, 4, 5]
autumn = [9, 10, 11]
df_month_help = pd.DataFrame(pd.to_datetime(df_tariff_1h.GMT).dt.month)
df_wea1h_spr3 = df_wea1h[df_month_help['GMT'] == 3]
df_wea1h_spr4 = df_wea1h[df_month_help['GMT'] == 4]
df_wea1h_spr5 = df_wea1h[df_month_help['GMT'] == 5]
df_wea1h_aut9 = df_wea1h[df_month_help['GMT'] == 9]
df_wea1h_aut10 = df_wea1h[df_month_help['GMT'] == 10]
df_wea1h_aut11 = df_wea1h[df_month_help['GMT'] == 11]

df_tou1h_spr3 = df_tou1h[df_month_help['GMT'] == 3]
df_tou1h_spr4 = df_tou1h[df_month_help['GMT'] == 4]
df_tou1h_spr5 = df_tou1h[df_month_help['GMT'] == 5]
df_tou1h_aut9 = df_tou1h[df_month_help['GMT'] == 9]
df_tou1h_aut10 = df_tou1h[df_month_help['GMT'] == 10]
df_tou1h_aut11 = df_tou1h[df_month_help['GMT'] == 11]

df_Ntou1h_spr3 = df_Ntou1h[df_month_help['GMT'] == 3]
df_Ntou1h_spr4 = df_Ntou1h[df_month_help['GMT'] == 4]
df_Ntou1h_spr5 = df_Ntou1h[df_month_help['GMT'] == 5]
df_Ntou1h_aut9 = df_Ntou1h[df_month_help['GMT'] == 9]
df_Ntou1h_aut10 = df_Ntou1h[df_month_help['GMT'] == 10]
df_Ntou1h_aut11 = df_Ntou1h[df_month_help['GMT'] == 11]
```

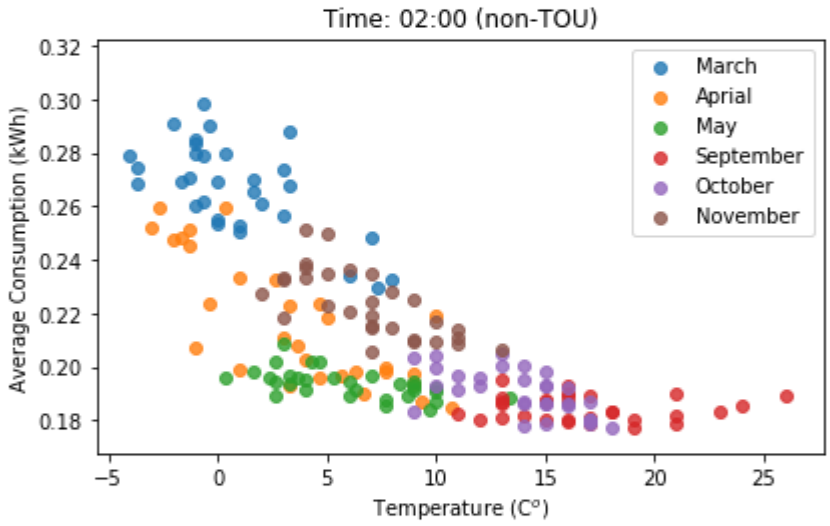
In [15]:
```
# Deeper look at Spring and Autumn
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
spring_autumn = [3, 4, 5, 9, 10, 11]
label_spr_aut = ['March', 'Aprial', 'May', 'September', 'October', 'November']
df_month_help = pd.DataFrame(pd.to_datetime(df_tariff_1h.GMT).dt.month)
# non-TOU vs TOU in terms of showing the elements in Average column such that they are hourly energy consumpt
ion v.s. temperature over 365 days
fig_all = plt.figure(figsize = (12,90))
ax_Ntou = [] # store subplot objects
ax_tou = [] # store subplot objects
lm = LinearRegression()
for i in range(24):
    ax_Ntou.append(fig_all.add_subplot(24, 2, 2 * i + 1))
    if i <= 9:
        for j in range(len(spring_autumn)):
            df_wea1h_month = df_wea1h[df_month_help['GMT'] == spring_autumn[j]]
            df_Ntou1h_month = df_Ntou1h[df_month_help['GMT'] == spring_autumn[j]]
            x = df_wea1h_month.TempC[df_wea1h_month.GMT.str.contains('0' + str(i) + ':00:00')].values
            y = df_Ntou1h_month.Average[df_Ntou1h_month.GMT.str.contains('0' + str(i) + ':00:00')].values
            ax_Ntou[-1].scatter(x, y, label = label_spr_aut[j], alpha = 0.8)
        ax_Ntou[-1].set_title('Time: 0' + str(i) + ':00 (non-TOU)')
        ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_Ntou[-1].set_ylabel('Average Consumption (kWh)')
        ax_Ntou[-1].legend()
    else:
        for j in range(len(spring_autumn)):
            df_wea1h_month = df_wea1h[df_month_help['GMT'] == spring_autumn[j]]
            df_Ntou1h_month = df_Ntou1h[df_month_help['GMT'] == spring_autumn[j]]
            x = df_wea1h_month.TempC[df_wea1h_month.GMT.str.contains(str(i) + ':00:00')].values
            y = df_Ntou1h_month.Average[df_Ntou1h_month.GMT.str.contains(str(i) + ':00:00')].values
            ax_Ntou[-1].scatter(x, y, label = label_spr_aut[j], alpha = 0.8)
        ax_Ntou[-1].set_title('Time: ' + str(i) + ':00 (non-TOU)')
        ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_Ntou[-1].set_ylabel('Average Consumption (kWh)')
        ax_Ntou[-1].legend()
for i in range(24):
    ax_tou.append(fig_all.add_subplot(24, 2, 2 * i + 2))
    if i <= 9:
        for j in range(len(spring_autumn)):
            df_wea1h_month = df_wea1h[df_month_help['GMT'] == spring_autumn[j]]
            df_tou1h_month = df_tou1h[df_month_help['GMT'] == spring_autumn[j]]
            x = df_wea1h_month.TempC[df_wea1h_month.GMT.str.contains('0' + str(i) + ':00:00')].values
            y = df_tou1h_month.Average[df_tou1h_month.GMT.str.contains('0' + str(i) + ':00:00')].values
            ax_tou[-1].scatter(x, y, label = label_spr_aut[j], alpha = 0.8)
        ax_tou[-1].set_title('Time: 0' + str(i) + ':00 (TOU)')
        ax_tou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_tou[-1].set_ylabel('Average Consumption (kWh)')
        ax_tou[-1].legend()
    else:
        for j in range(len(spring_autumn)):
            df_wea1h_month = df_wea1h[df_month_help['GMT'] == spring_autumn[j]]
            df_tou1h_month = df_tou1h[df_month_help['GMT'] == spring_autumn[j]]
            x = df_wea1h_month.TempC[df_wea1h_month.GMT.str.contains(str(i) + ':00:00')].values
            y = df_tou1h_month.Average[df_tou1h_month.GMT.str.contains(str(i) + ':00:00')].values
            ax_tou[-1].scatter(x, y, label = label_spr_aut[j], alpha = 0.8)
        ax_tou[-1].set_title('Time: ' + str(i) + ':00 (TOU)')
        ax_tou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_tou[-1].set_ylabel('Average Consumption (kWh)')
        ax_tou[-1].legend()
plt.tight_layout()
```

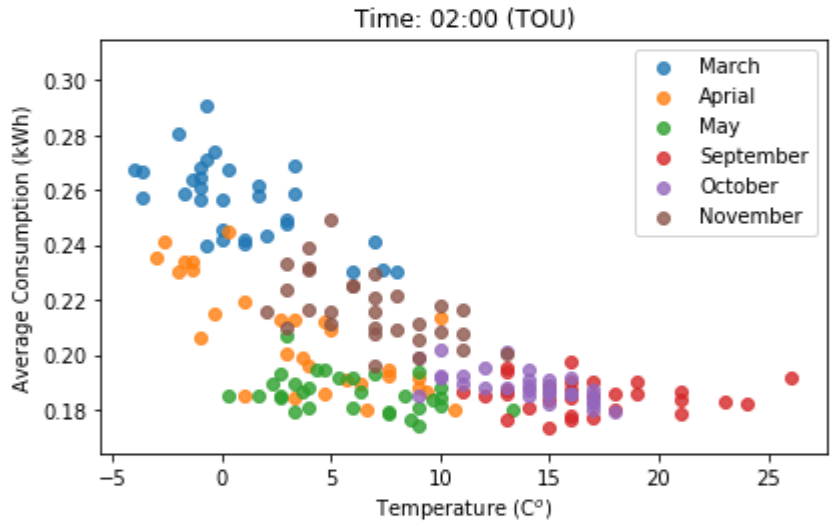

Time: 00:00 (non-TOU)
Average Consumption (kWh)
Temperature (C°)
March
Aprial
May
September
October
November

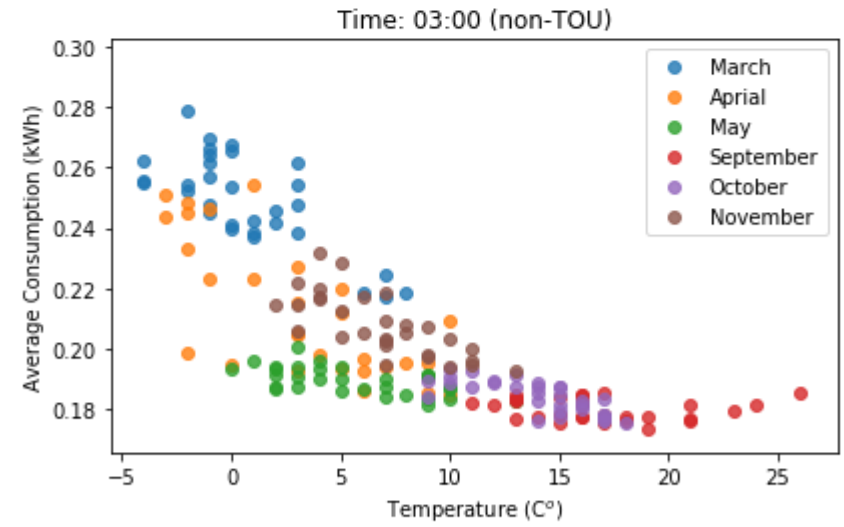

Time: 00:00 (TOU)
Average Consumption (kWh)
Temperature (C°)
March
Aprial
May
September
October
November

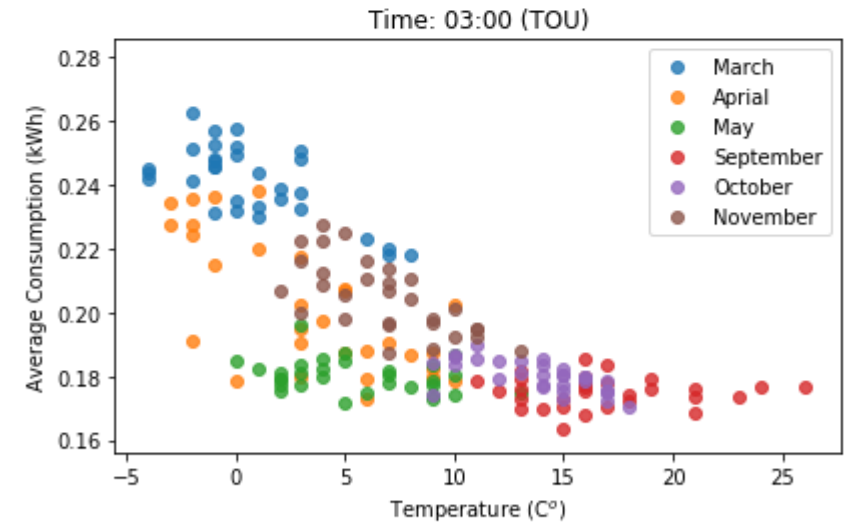

Time: 01:00 (non-TOU)
Average Consumption (kWh)
Temperature (C°)
March
Aprial
May
September
October
November

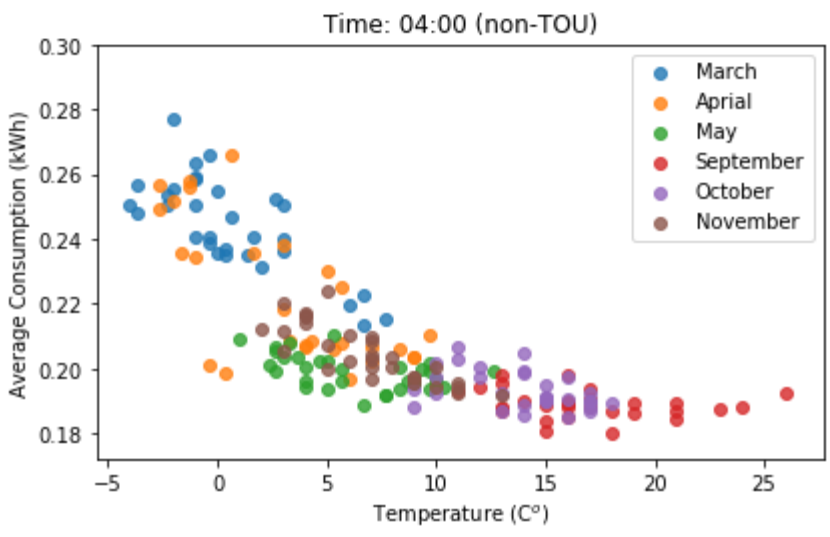

Time: 01:00 (TOU)
Average Consumption (kWh)
Temperature (C°)
March
Aprial
May
September
October
November

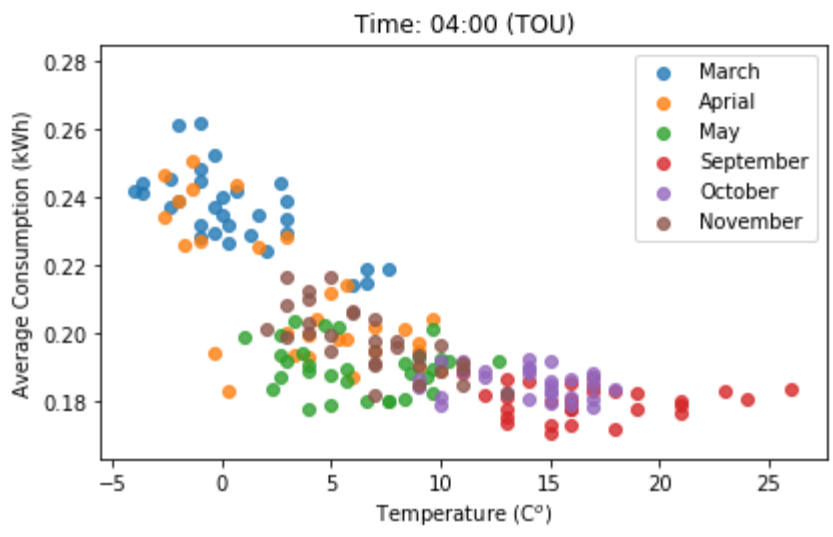

Time: 02:00 (non-TOU)
Average Consumption (kWh)
Temperature (C°)
March
Aprial
May
September
October
November


Time: 02:00 (TOU)
Average Consumption (kWh)
Temperature (C°)
March
Aprial
May
September
October
November


Time: 03:00 (non-TOU)
Average Consumption (kWh)
Temperature (C°)
March
Aprial
May
September
October
November

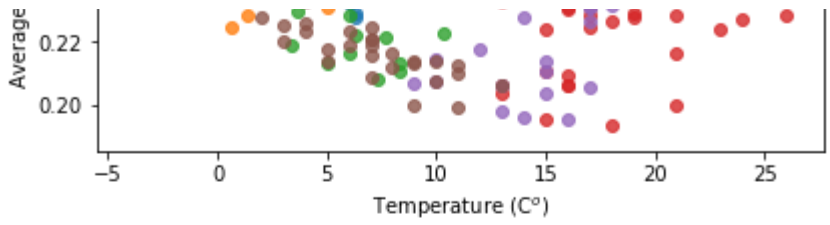

Time: 03:00 (TOU)
Average Consumption (kWh)
Temperature (C°)
March
Aprial
May
September
October
November

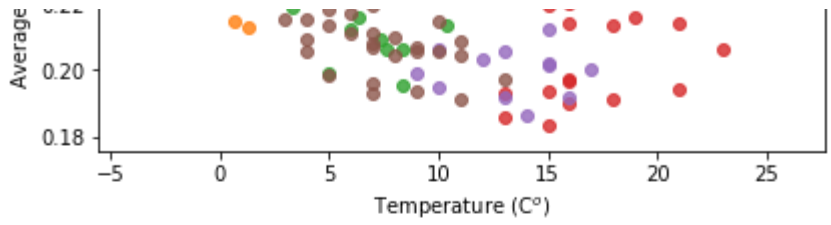

Time: 04:00 (non-TOU)
Average Consumption (kWh)
Temperature (C°)
March
Aprial
May
September
October
November

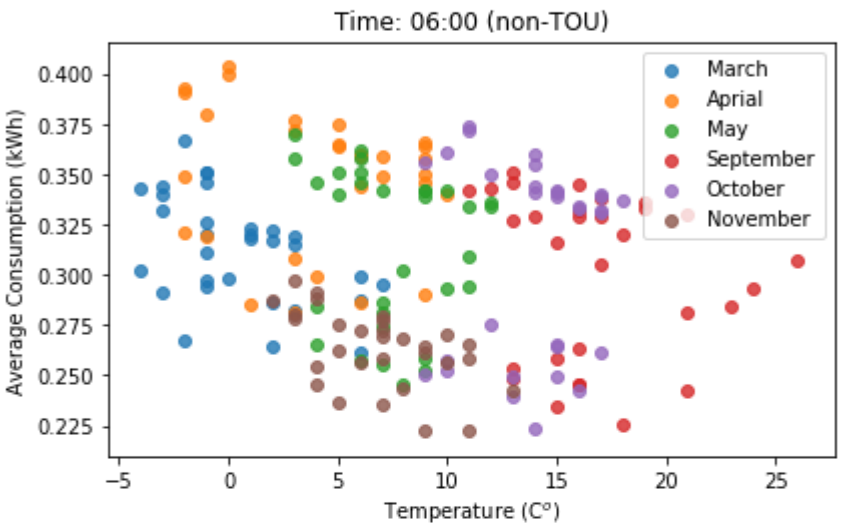

Time: 04:00 (TOU)
Average Consumption (kWh)
Temperature (C°)
March
Aprial
May
September
October
November

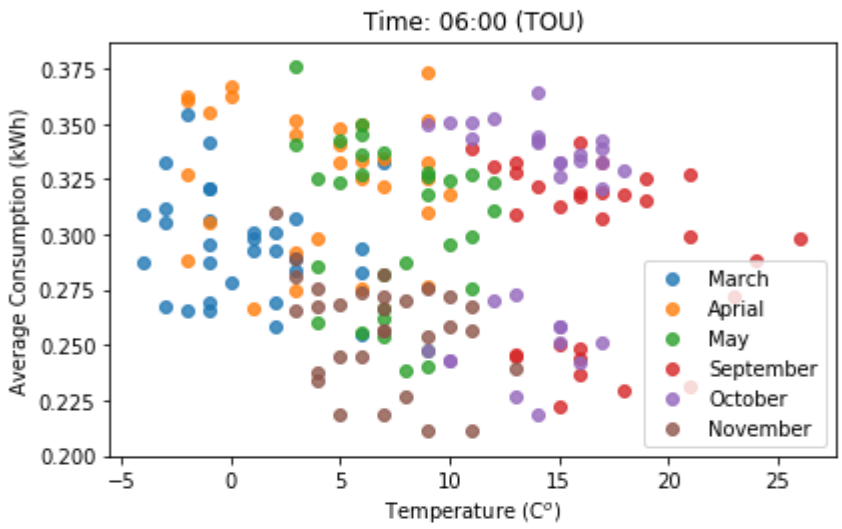
Time: 05:00 (non-TOU)
Average Consumption (kWh)
Temperature (C°)
March
Aprial
May
September
October
November

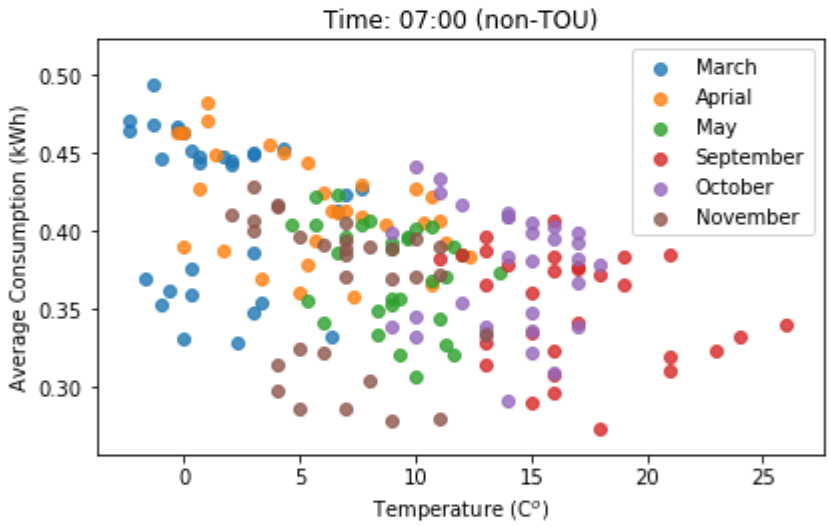
Time: 05:00 (TOU)
Average Consumption (kWh)
Temperature (C°)
March
Aprial
May
September
October
November

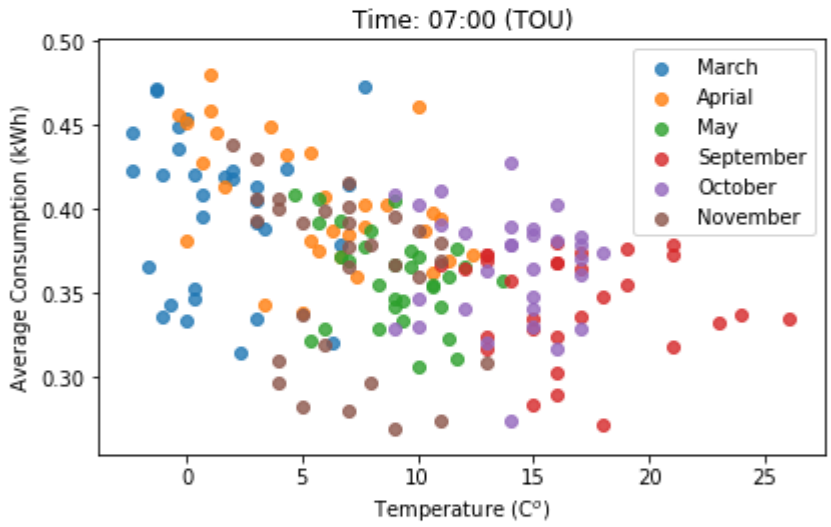

Average
0.22
0.20
−5
0
5
10
15
20
25
Temperature (C°)

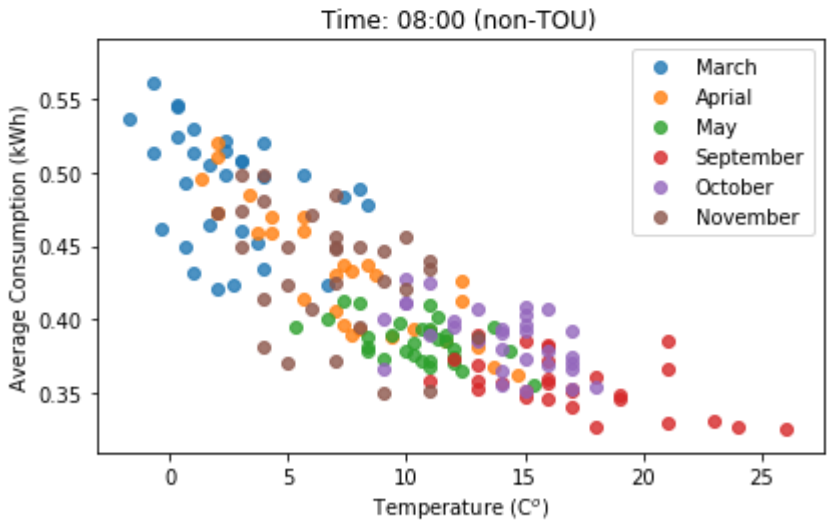

Average
0.20
0.18
−5
0
5
10
15
20
25
Temperature (C°)

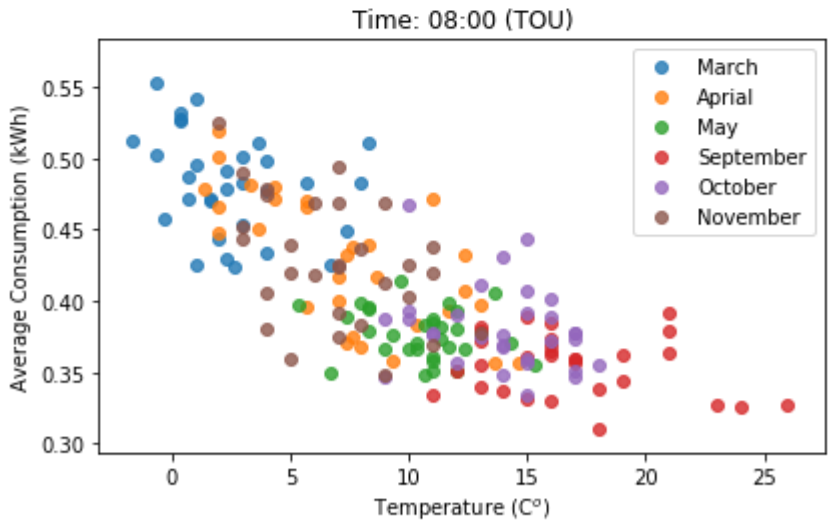

Time: 06:00 (non-TOU)
Average Consumption (kWh)
0.400
0.375
0.350
0.325
0.300
0.275
0.250
0.225
March
Aprial
May
September
October
November
−5
0
5
10
15
20
25
Temperature (C°)

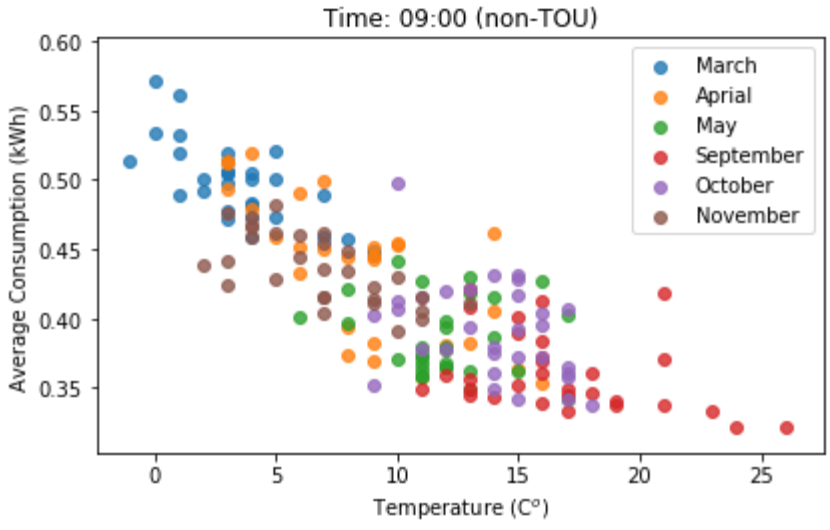

Time: 06:00 (TOU)
Average Consumption (kWh)
0.375
0.350
0.325
0.300
0.275
0.250
0.225
0.200
March
Aprial
May
September
October
November
−5
0
5
10
15
20
25
Temperature (C°)

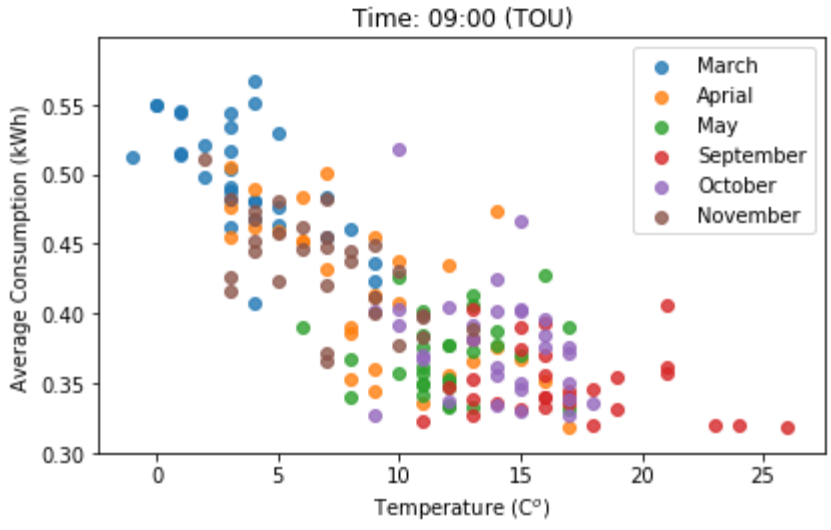

Time: 07:00 (non-TOU)
Average Consumption (kWh)
0.50
0.45
0.40
0.35
0.30
March
Aprial
May
September
October
November
0
5
10
15
20
25
Temperature (C°)

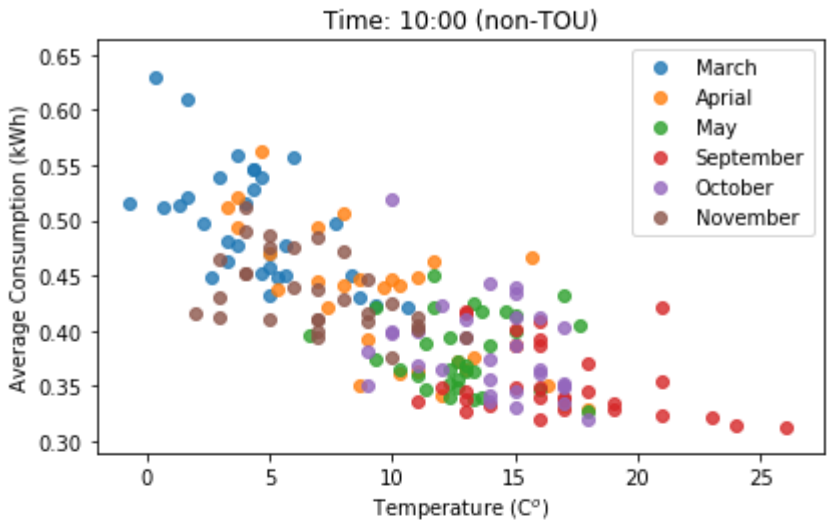

Time: 07:00 (TOU)
Average Consumption (kWh)
0.50
0.45
0.40
0.35
0.30
March
Aprial
May
September
October
November
0
5
10
15
20
25
Temperature (C°)

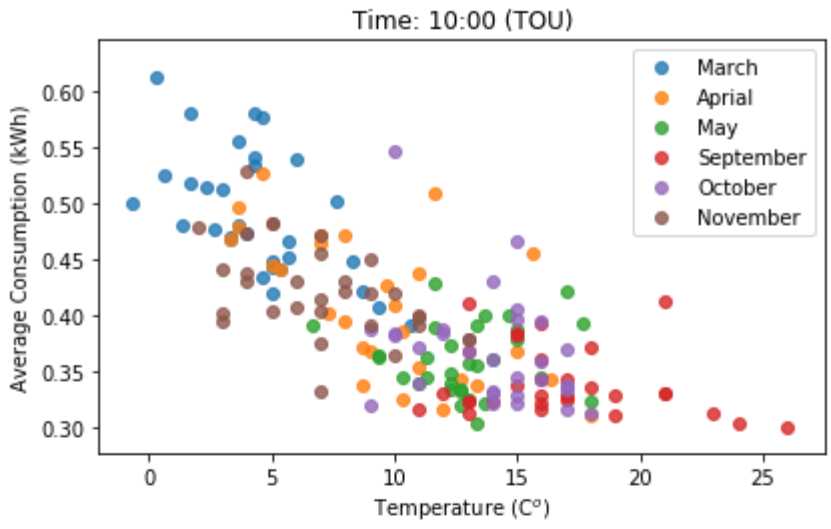

Time: 08:00 (non-TOU)
Average Consumption (kWh)
0.55
0.50
0.45
0.40
0.35
March
Aprial
May
September
October
November
0
5
10
15
20
25
Temperature (C°)

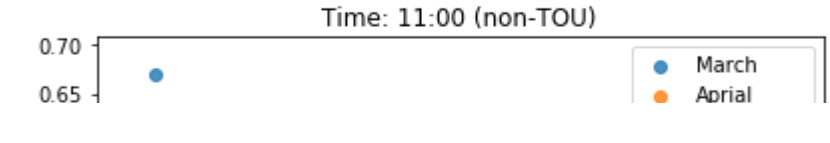

Time: 08:00 (TOU)
Average Consumption (kWh)
0.55
0.50
0.45
0.40
0.35
0.30
March
Aprial
May
September
October
November
0
5
10
15
20
25
Temperature (C°)

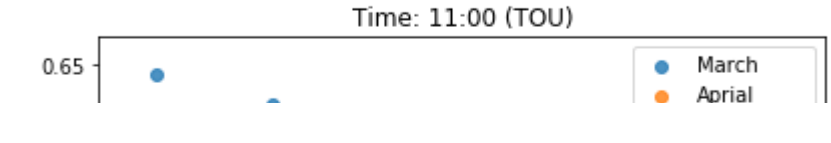

Time: 09:00 (non-TOU)
Average Consumption (kWh)
0.60
0.55
0.50
0.45
0.40
0.35
March
Aprial
May
September
October
November
0
5
10
15
20
25
Temperature (C°)

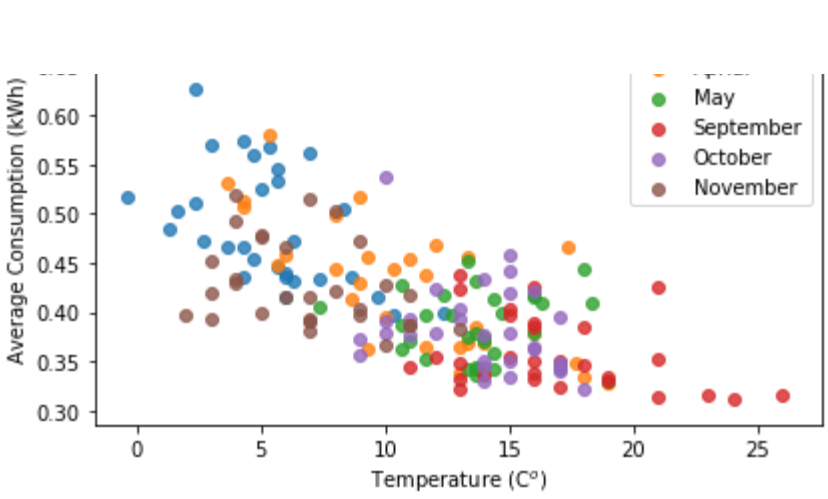

Time: 09:00 (TOU)
Average Consumption (kWh)
0.55
0.50
0.45
0.40
0.35
0.30
March
Aprial
May
September
October
November
0
5
10
15
20
25
Temperature (C°)

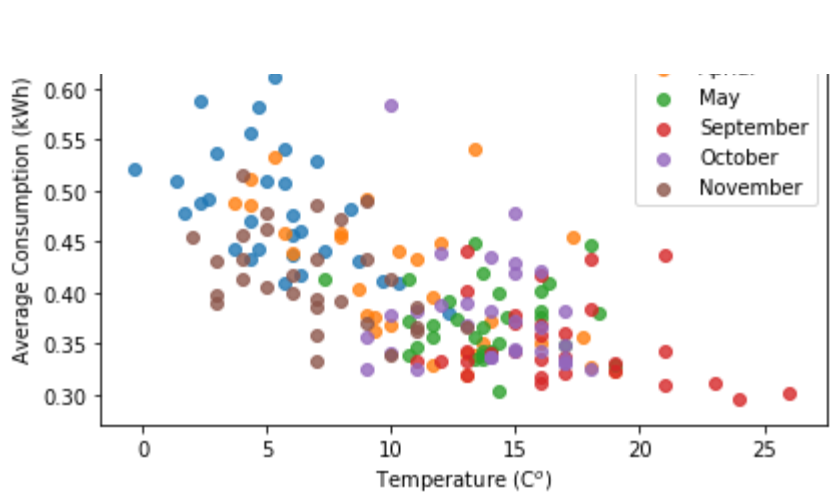

Time: 10:00 (non-TOU)
Average Consumption (kWh)
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
March
Aprial
May
September
October
November
0
5
10
15
20
25
Temperature (C°)

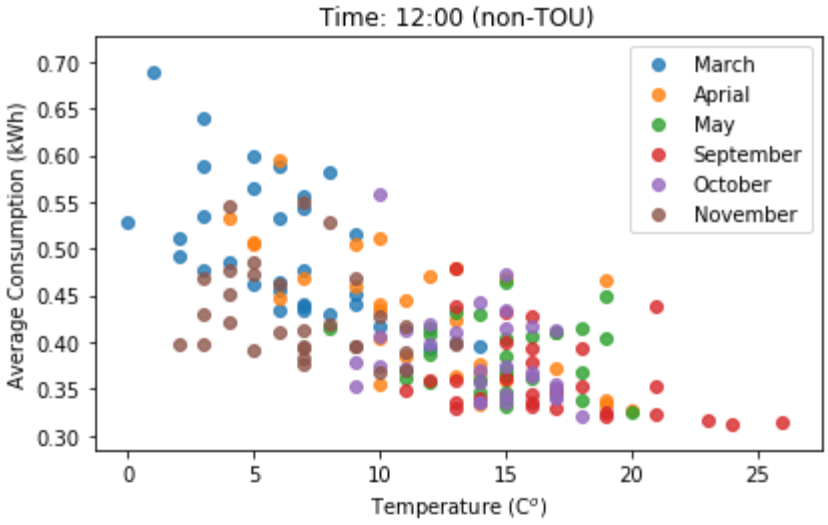

Time: 10:00 (TOU)
Average Consumption (kWh)
0.60
0.55
0.50
0.45
0.40
0.35
0.30
March
Aprial
May
September
October
November
0
5
10
15
20
25
Temperature (C°)

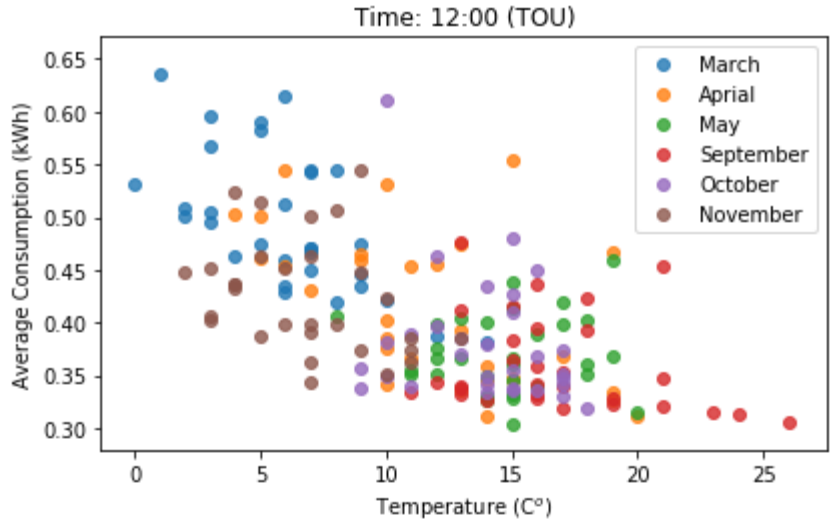

Time: 11:00 (non-TOU)
0.70
0.65
March
Aprial

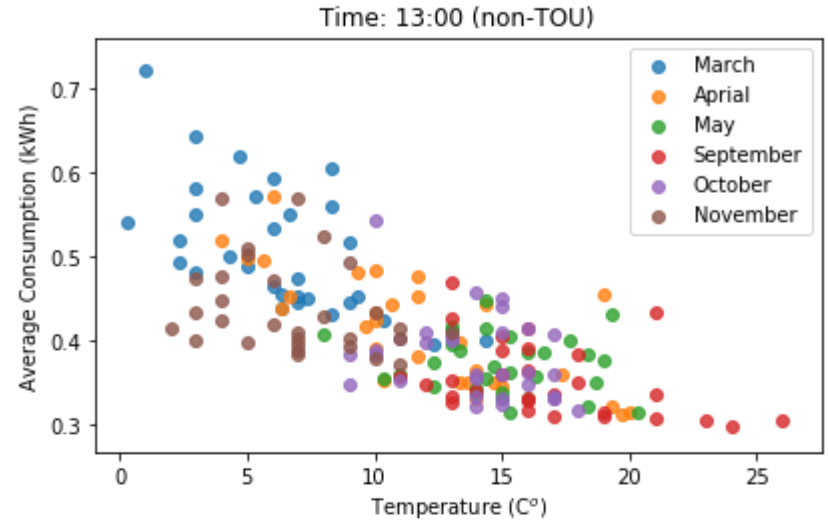

Time: 11:00 (TOU)
0.65
March
Aprial

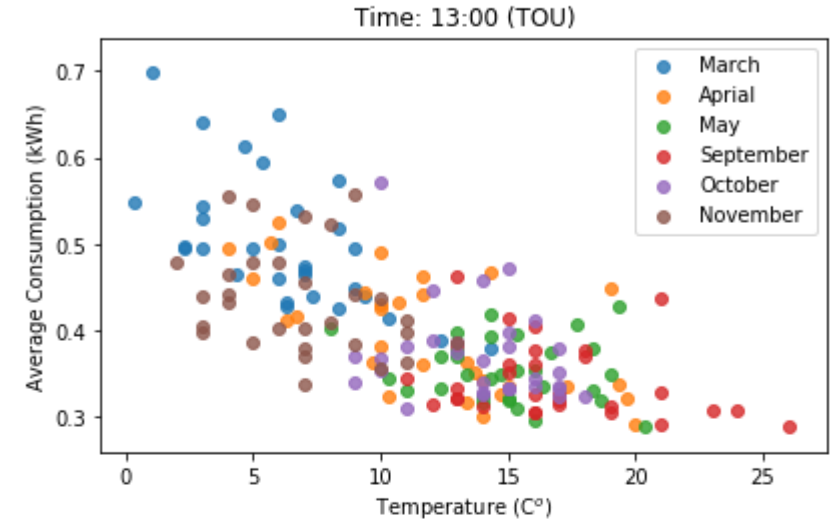

May
September
October
November
Average Consumption (kWh)
Temperature (C°)

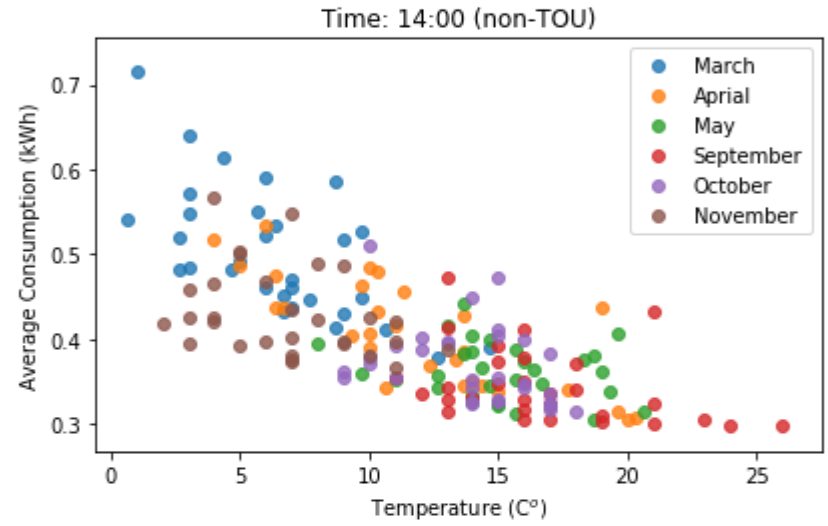

May
September
October
November
Average Consumption (kWh)
Temperature (C°)

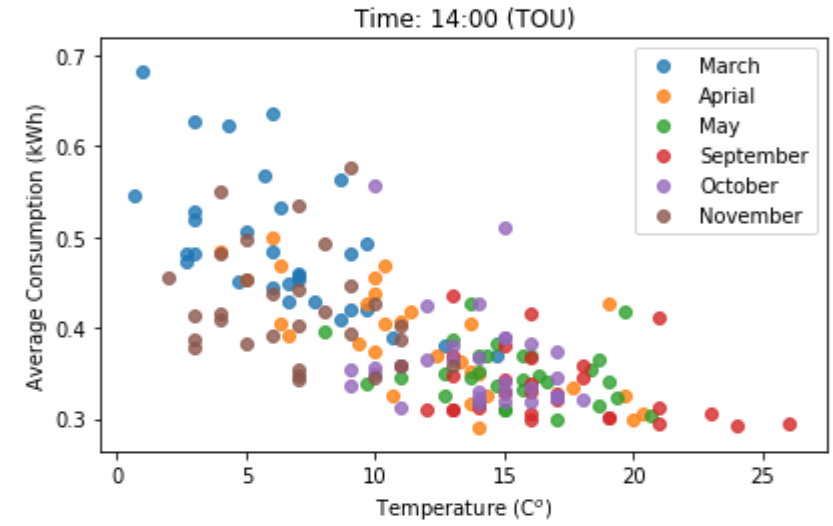

Time: 12:00 (non-TOU)
March
Aprial
May
September
October
November
Average Consumption (kWh)
Temperature (C°)

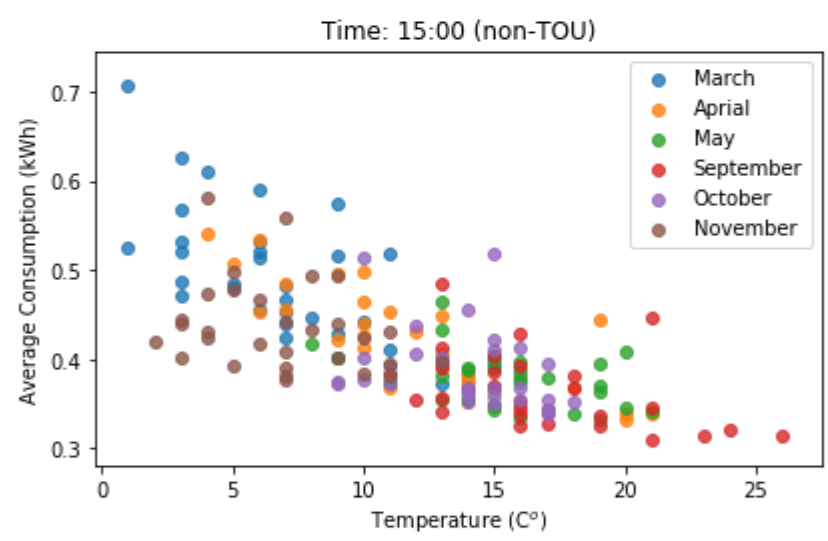

Time: 12:00 (TOU)
March
Aprial
May
September
October
November
Average Consumption (kWh)
Temperature (C°)

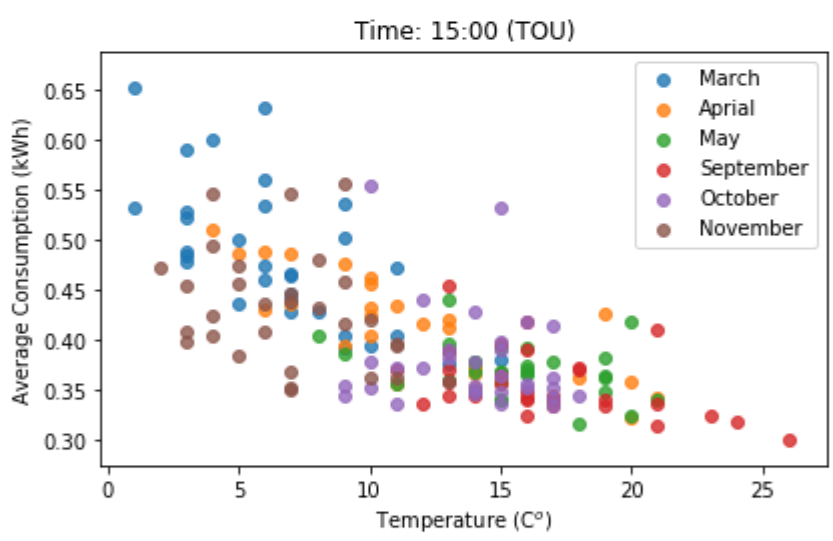

Time: 13:00 (non-TOU)
March
Aprial
May
September
October
November
Average Consumption (kWh)
Temperature (C°)

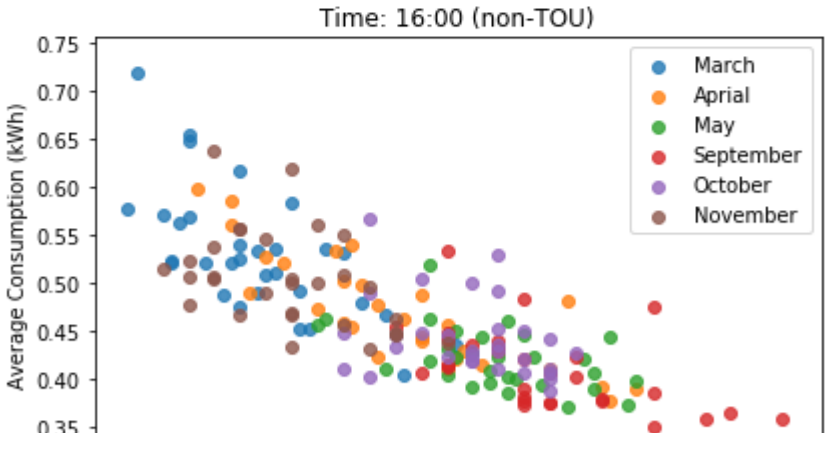

Time: 13:00 (TOU)
March
Aprial
May
September
October
November
Average Consumption (kWh)
Temperature (C°)

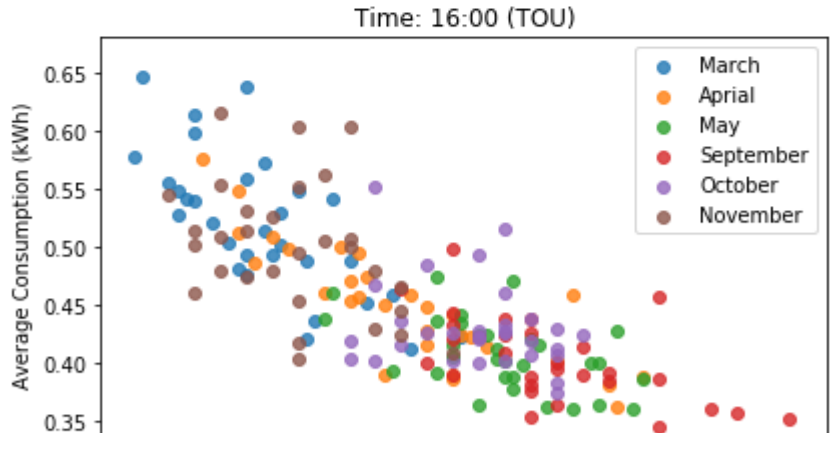

Time: 14:00 (non-TOU)
March
Aprial
May
September
October
November
Average Consumption (kWh)
Temperature (C°)

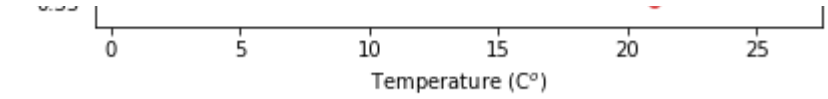

Time: 14:00 (TOU)
March
Aprial
May
September
October
November
Average Consumption (kWh)
Temperature (C°)

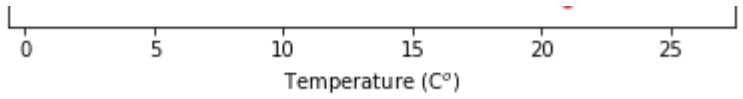

Time: 15:00 (non-TOU)
March
Aprial
May
September
October
November
Average Consumption (kWh)
Temperature (C°)

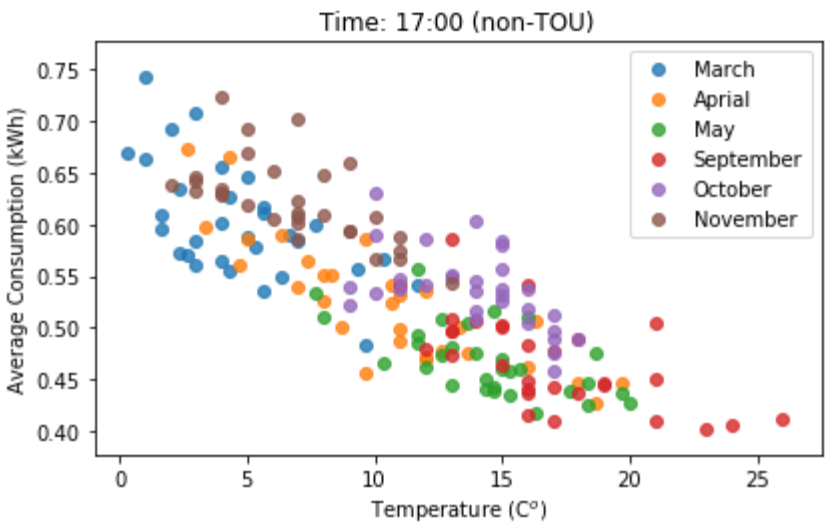

Time: 15:00 (TOU)
March
Aprial
May
September
October
November
Average Consumption (kWh)
Temperature (C°)

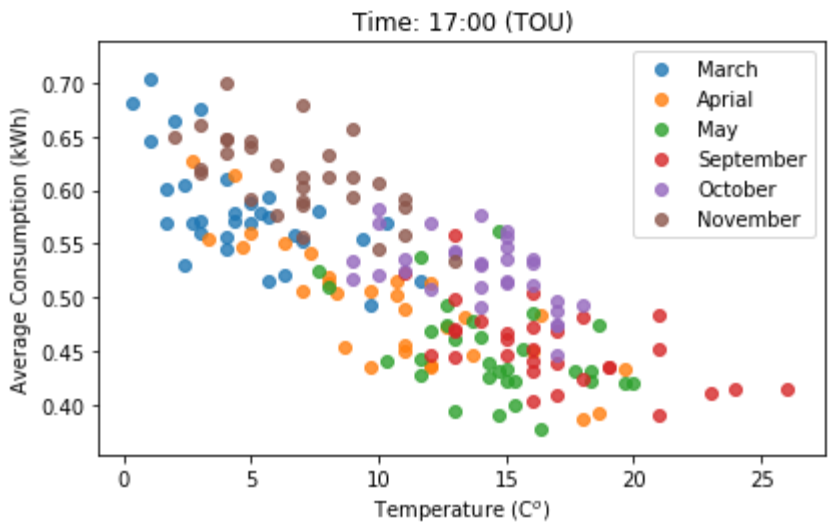

Time: 16:00 (non-TOU)
March
Aprial
May
September
October
November
Average Consumption (kWh)

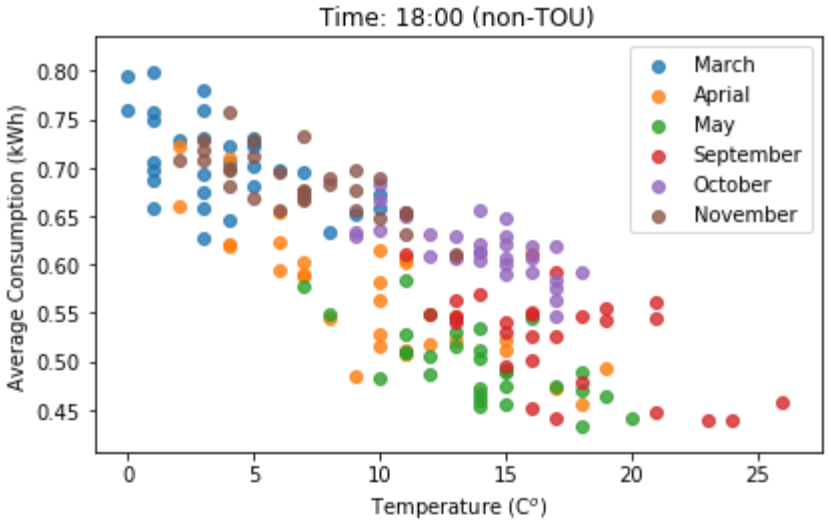

Time: 16:00 (TOU)
March
Aprial
May
September
October
November
Average Consumption (kWh)

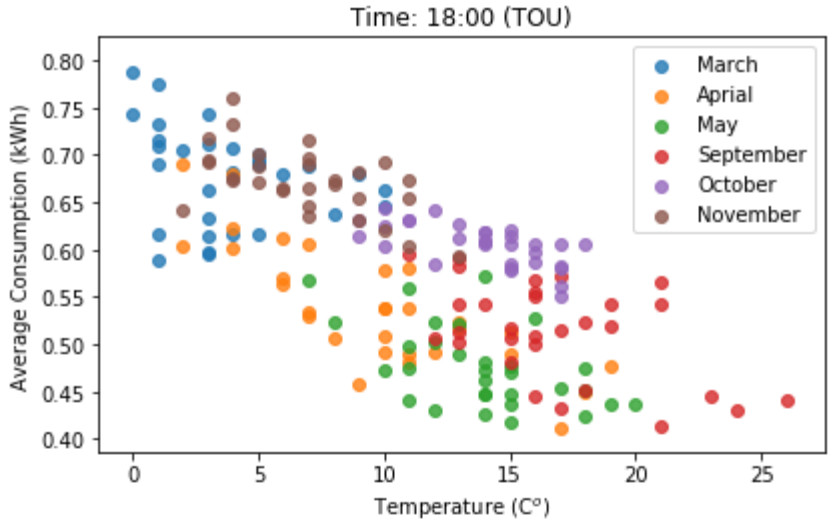

0
5
10
15
20
25
Temperature (C°)

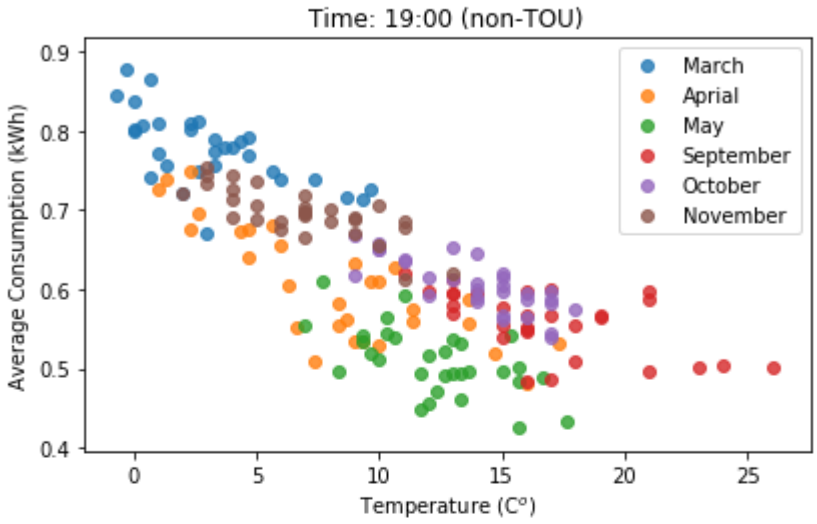

0
5
10
15
20
25
Temperature (C°)

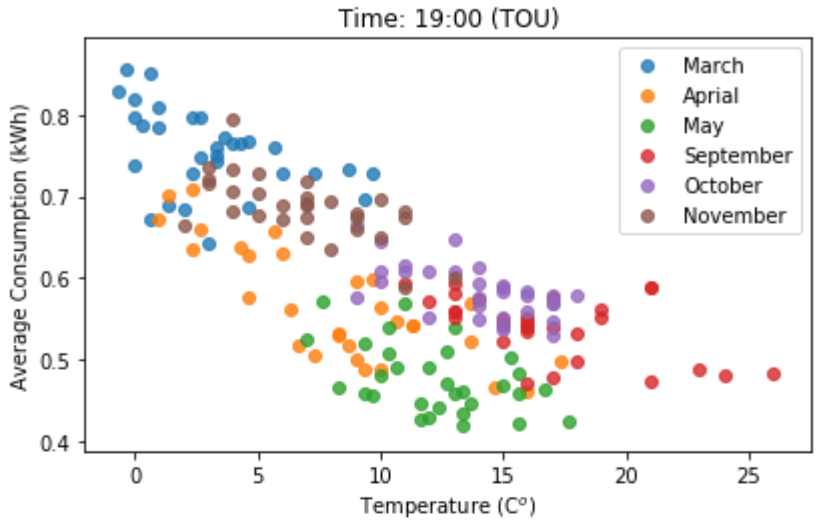

Time: 17:00 (non-TOU)
Average Consumption (kWh)
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0
5
10
15
20
25
Temperature (C°)
March
Aprial
May
September
October
November

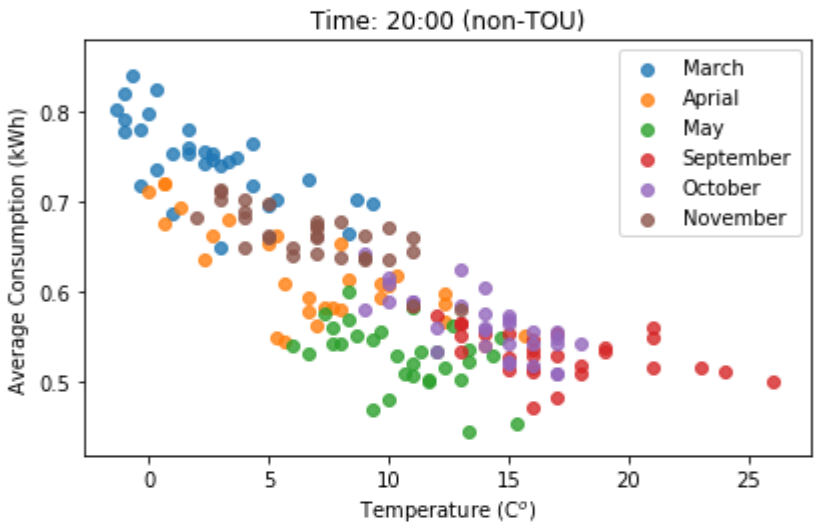

Time: 17:00 (TOU)
Average Consumption (kWh)
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0
5
10
15
20
25
Temperature (C°)
March
Aprial
May
September
October
November

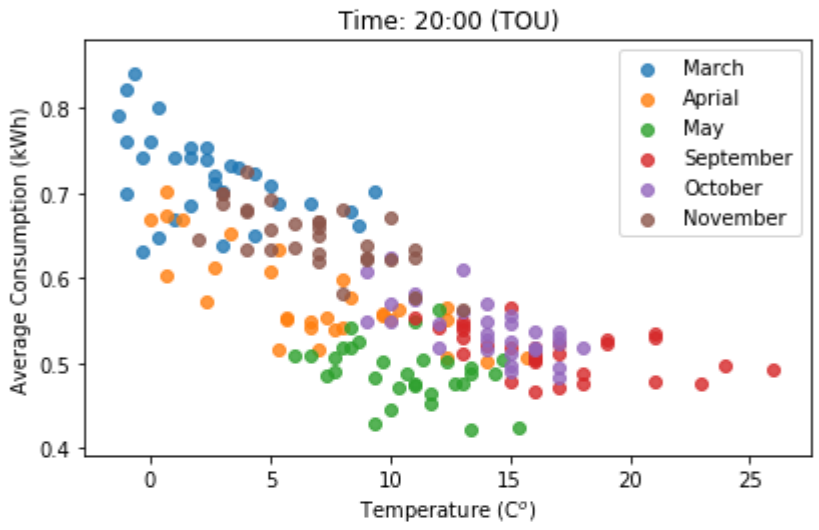

Time: 18:00 (non-TOU)
Average Consumption (kWh)
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0
5
10
15
20
25
Temperature (C°)
March
Aprial
May
September
October
November

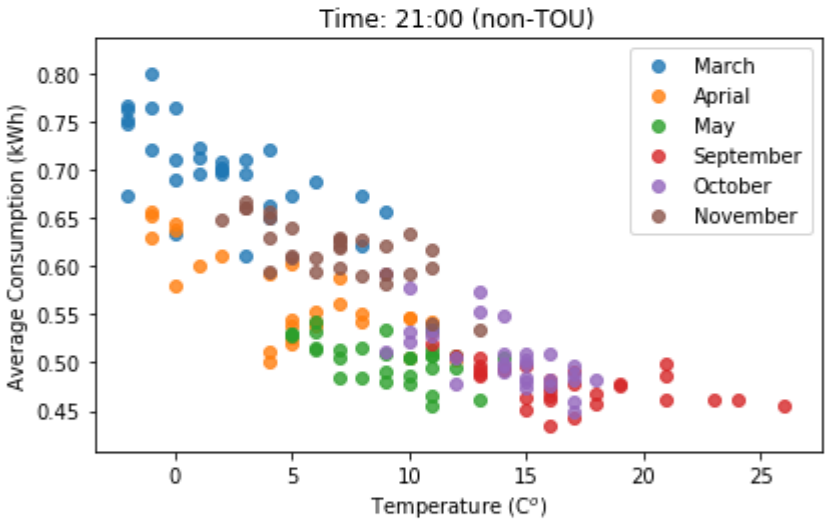

Time: 18:00 (TOU)
Average Consumption (kWh)
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0
5
10
15
20
25
Temperature (C°)
March
Aprial
May
September
October
November

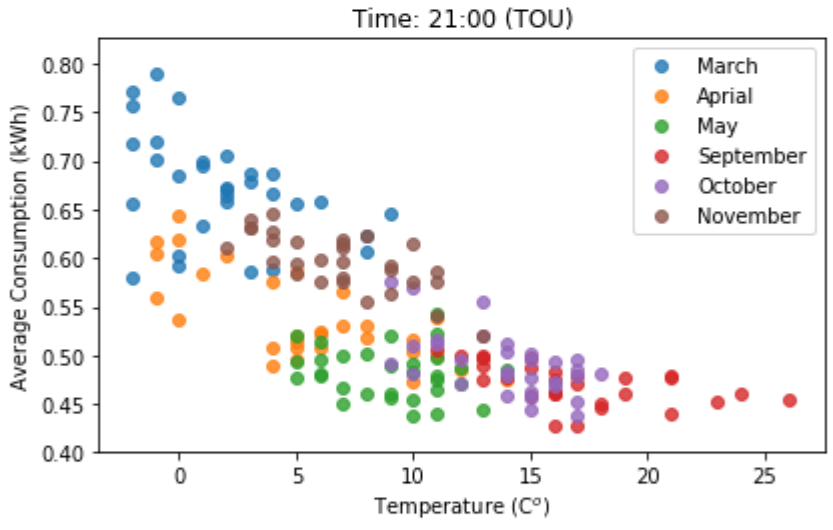

Time: 19:00 (non-TOU)
Average Consumption (kWh)
0.9
0.8
0.7
0.6
0.5
0.4
0
5
10
15
20
25
Temperature (C°)
March
Aprial
May
September
October
November

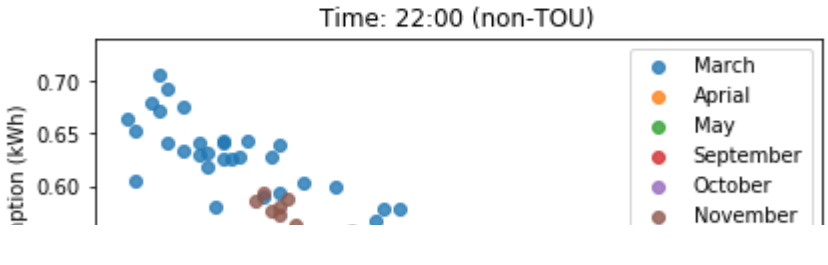

Time: 19:00 (TOU)
Average Consumption (kWh)
0.8
0.7
0.6
0.5
0.4
0
5
10
15
20
25
Temperature (C°)
March
Aprial
May
September
October
November

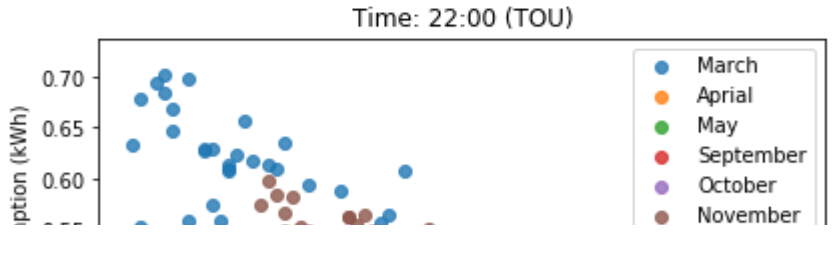

Time: 20:00 (non-TOU)
Average Consumption (kWh)
0.8
0.7
0.6
0.5
0
5
10
15
20
25
Temperature (C°)
March
Aprial
May
September
October
November

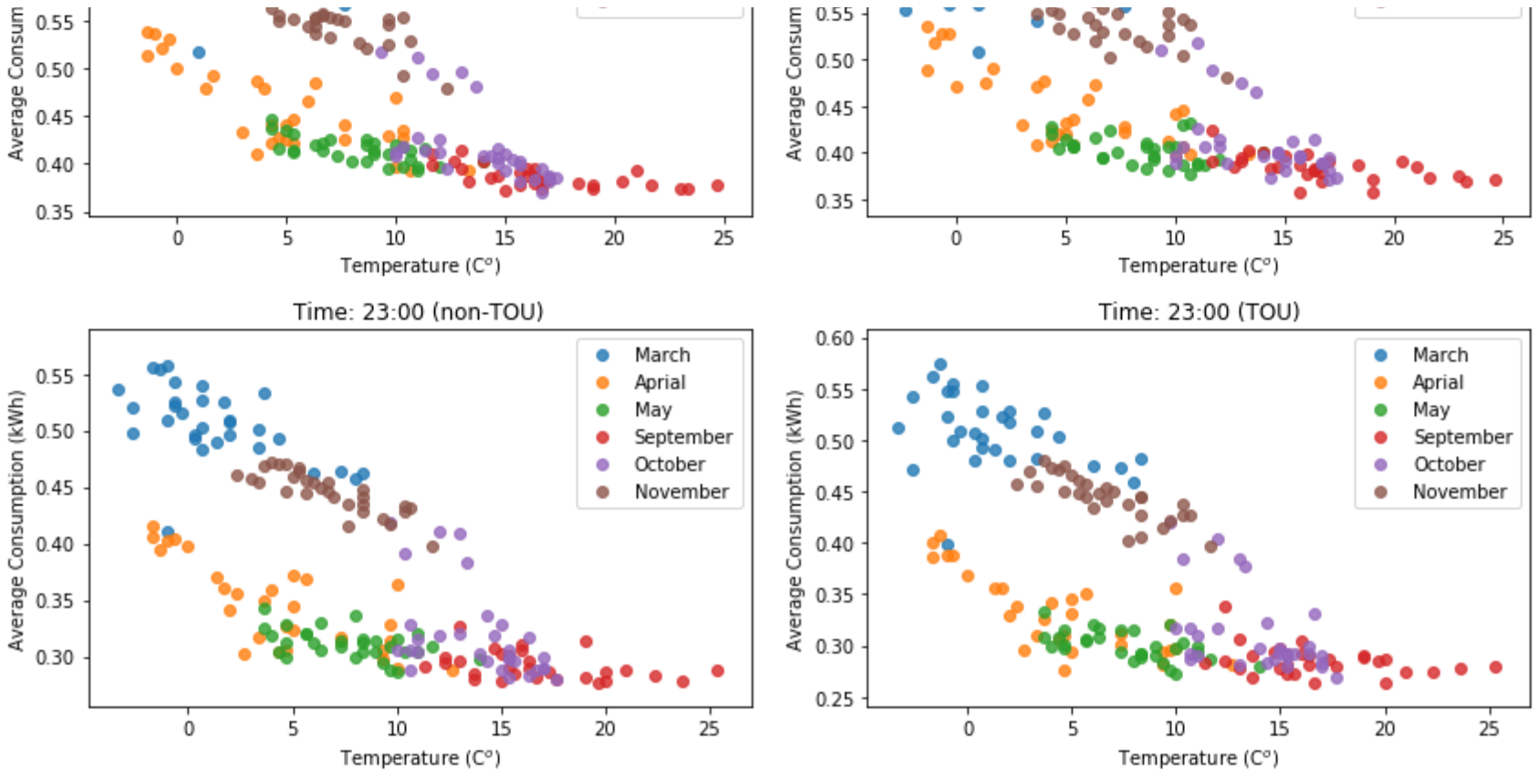

Time: 20:00 (TOU)
Average Consumption (kWh)
0.8
0.7
0.6
0.5
0.4
0
5
10
15
20
25
Temperature (C°)
March
Aprial
May
September
October
November


Time: 21:00 (non-TOU)
Average Consumption (kWh)
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0
5
10
15
20
25
Temperature (C°)
March
Aprial
May
September
October
November


Time: 21:00 (TOU)
Average Consumption (kWh)
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0
5
10
15
20
25
Temperature (C°)
March
Aprial
May
September
October
November


Time: 22:00 (non-TOU)
0.70
0.65
0.60
March
Aprial
May
September
October
November


Time: 22:00 (TOU)
0.70
0.65
0.60
March
Aprial
May
September
October
November



The above shows that the left upper parts of the pictures are always from November (Autumn) and March (Spring)

April & May (Spring) and Sep & Oct. (Autumn) are on the lower part of the pictures.

November, March and Winter (Dec, Jan, Feb.), i.e., months of 11, 12, 1, 2, 3 are on the upper left part.

April & May, Sep & Oct, and Summer (Jun, July, August), i.e., months of 4, 5, 6, 7, 8, 9, 10 are on the lower (sometimes right) part

Monthly average demand and monthly total demand

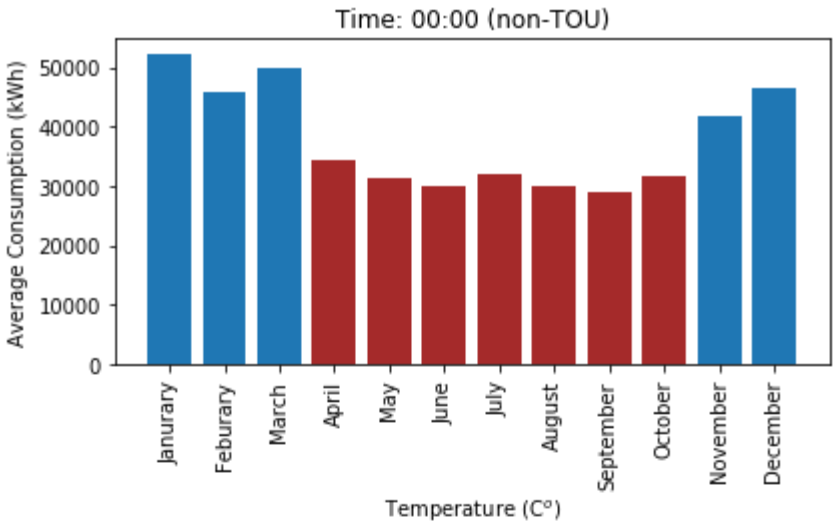
In [18]:
```
df_Ntou1h_month.Total.sum()
```

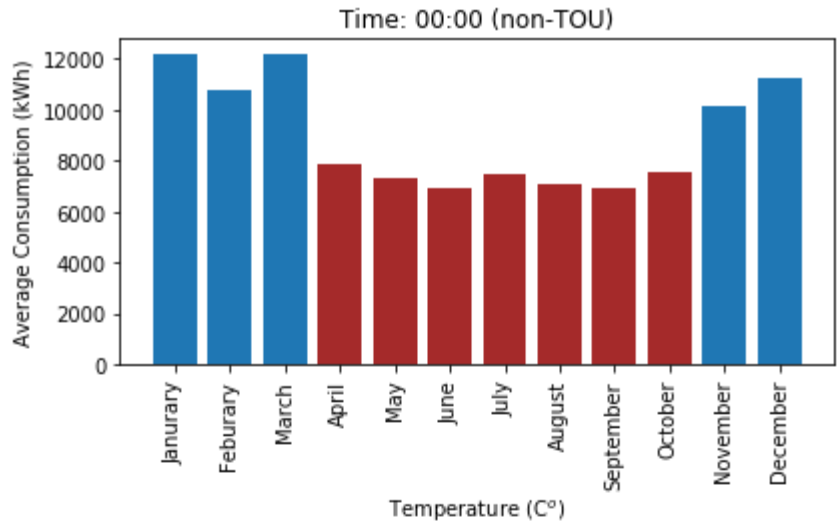
Out[18]: 1292443.6039999998

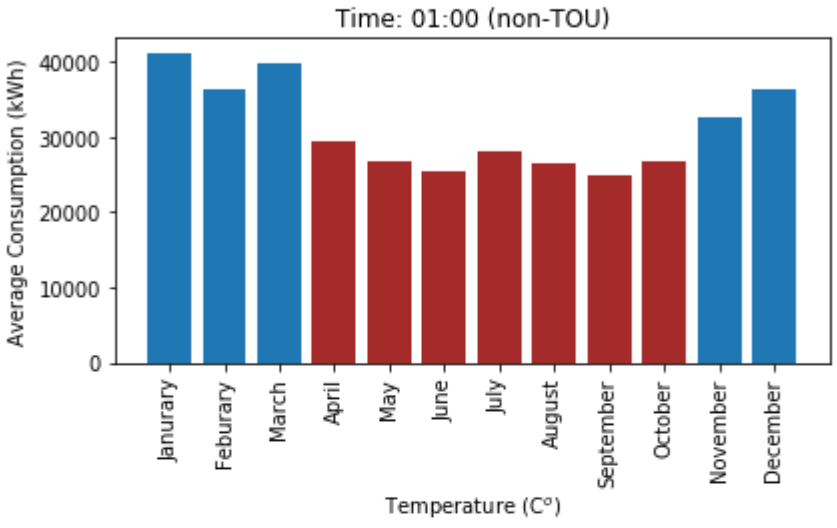
In [20]:
```
df_Ntou1h_month.Total[df_Ntou1h_month.GMT.str.contains('0' + str(i) + ':00:00')].sum()
```

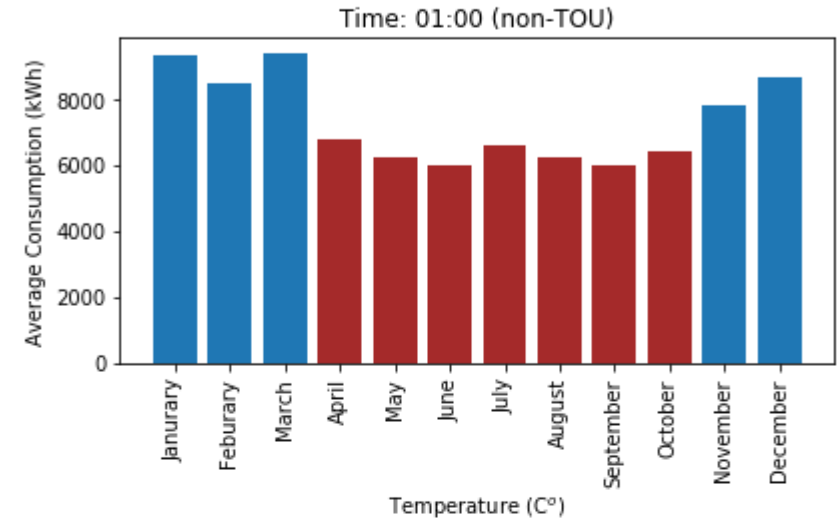
Out[20]: 52187.92

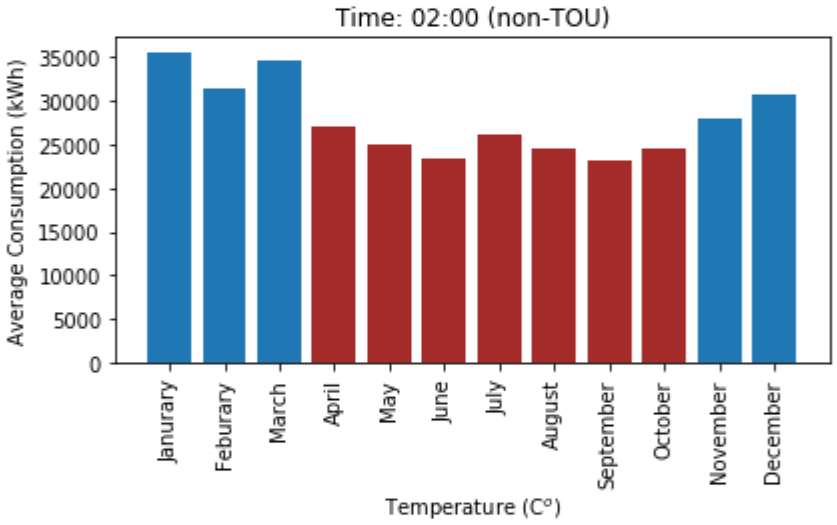
In [47]:
```
# Monthly total demand
month_n = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
month_list = ['Janurary', 'Feburary', 'March', 'April', 'May', 'June',
              'July', 'August', 'September', 'October', 'November', 'December']
color_dif = ['C0', 'C0', 'C0', 'brown', 'brown', 'brown',
             'brown', 'brown', 'brown', 'brown', 'C0', 'C0']
df_month_help = pd.DataFrame(pd.to_datetime(df_tariff_1h.GMT).dt.month)
# non-TOU vs TOU in terms of showing the elements in Average column such that they are hourly energy consumpt
ion v.s. temperature over 365 days
fig_all = plt.figure(figsize = (12,90))
ax_Ntou = [] # store subplot objects
ax_tou = [] # store subplot objects
lm = LinearRegression()
for i in range(24):
    ax_Ntou.append(fig_all.add_subplot(24, 2, 2 * i + 1))
    if i <= 9:
        month_hour_tot = []
        for j in range(len(month_n)):
            df_Ntou1h_month = df_Ntou1h[df_month_help['GMT'] == month_n[j]]
            month_hour_tot.append(df_Ntou1h_month.Total[df_Ntou1h_month.GMT.str.contains('0' + str(i) + ':00:
00')].sum())
        ax_Ntou[-1].bar(range(len(month_n)), month_hour_tot, color = color_dif)
        ax_Ntou[-1].set_xticks(range(len(month_n)))
        ax_Ntou[-1].set_xticklabels(month_list, rotation = 90)
        ax_Ntou[-1].set_title('Time: 0' + str(i) + ':00 (non-TOU)')
        ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_Ntou[-1].set_ylabel('Average Consumption (kWh)')
    else:
        month_hour_tot = []
        for j in range(len(month_n)):
            df_Ntou1h_month = df_Ntou1h[df_month_help['GMT'] == month_n[j]]
            month_hour_tot.append(df_Ntou1h_month.Total[df_Ntou1h_month.GMT.str.contains(str(i) + ':00:00')].
sum())
        ax_Ntou[-1].bar(range(len(month_n)), month_hour_tot, color = color_dif)
        ax_Ntou[-1].set_xticks(range(len(month_n)))
        ax_Ntou[-1].set_xticklabels(month_list, rotation = 90)
        ax_Ntou[-1].set_title('Time: ' + str(i) + ':00 (non-TOU)')
        ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_Ntou[-1].set_ylabel('Average Consumption (kWh)')
for i in range(24):
    ax_tou.append(fig_all.add_subplot(24, 2, 2 * i + 2))
    if i <= 9:
        month_hour_tot = []
        for j in range(len(month_n)):
            df_tou1h_month = df_tou1h[df_month_help['GMT'] == month_n[j]]
            month_hour_tot.append(df_tou1h_month.Total[df_tou1h_month.GMT.str.contains('0' + str(i) + ':00:0
0')].sum())
        ax_tou[-1].bar(range(len(month_n)), month_hour_tot, color = color_dif)
        ax_tou[-1].set_xticks(range(len(month_n)))
        ax_tou[-1].set_xticklabels(month_list, rotation = 90)
        ax_tou[-1].set_title('Time: 0' + str(i) + ':00 (non-TOU)')
        ax_tou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_tou[-1].set_ylabel('Average Consumption (kWh)')
    else:
        month_hour_tot = []
        for j in range(len(month_n)):
            df_tou1h_month = df_tou1h[df_month_help['GMT'] == month_n[j]]
            month_hour_tot.append(df_tou1h_month.Total[df_tou1h_month.GMT.str.contains(str(i) + ':00:00')].su
m())
        ax_tou[-1].bar(range(len(month_n)), month_hour_tot, color = color_dif)
        ax_tou[-1].set_xticks(range(len(month_n)))
        ax_tou[-1].set_xticklabels(month_list, rotation = 90)
        ax_tou[-1].set_title('Time: ' + str(i) + ':00 (non-TOU)')
        ax_tou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_tou[-1].set_ylabel('Average Consumption (kWh)')
plt.tight_layout()
```

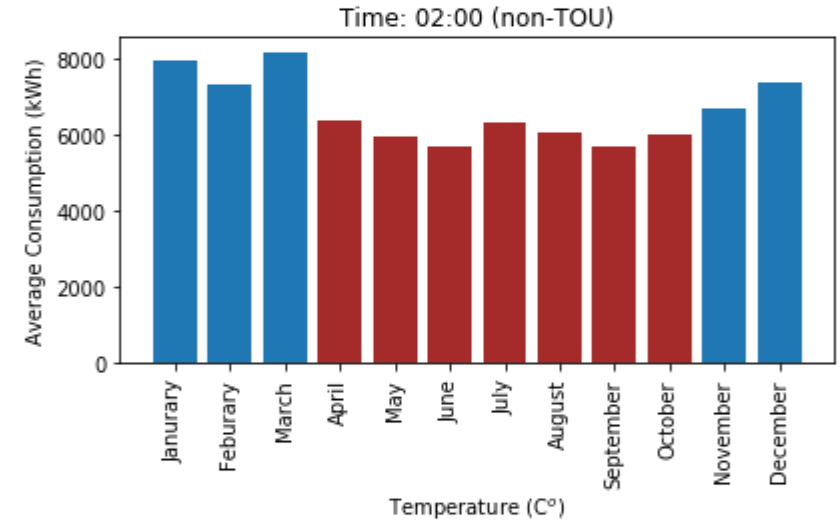

Time: 00:00 (non-TOU)
Average Consumption (kWh)
0
10000
20000
30000
40000
50000
Janurary
Feburary
March
April
May
June
July
August
September
October
November
December
Temperature (C°)

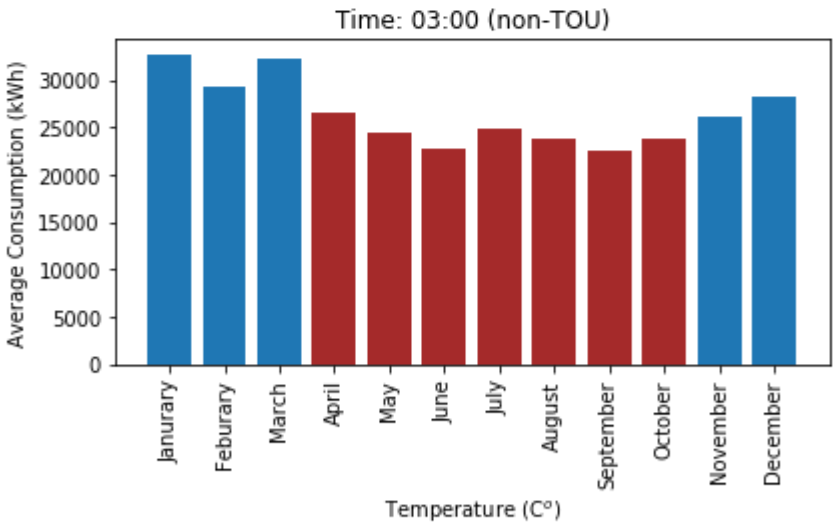

Time: 00:00 (non-TOU)
Average Consumption (kWh)
0
2000
4000
6000
8000
10000
12000
Janurary
Feburary
March
April
May
June
July
August
September
October
November
December
Temperature (C°)

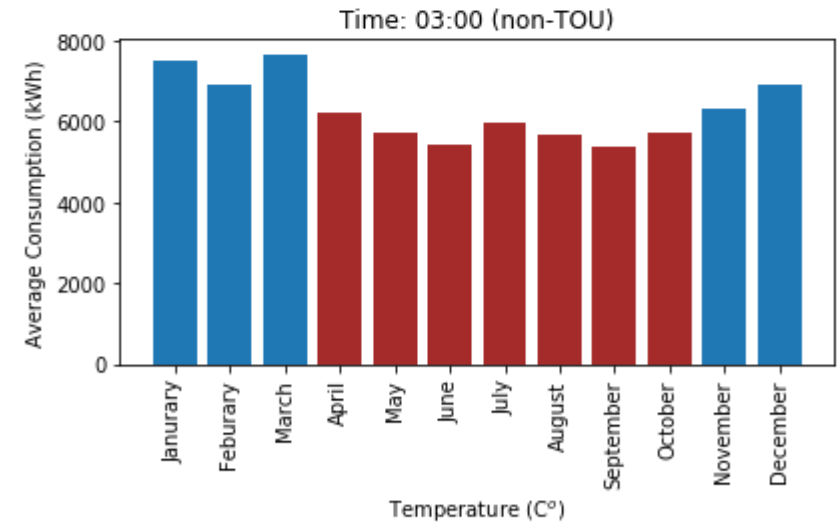

Time: 01:00 (non-TOU)
Average Consumption (kWh)
0
10000
20000
30000
40000
Janurary
Feburary
March
April
May
June
July
August
September
October
November
December
Temperature (C°)

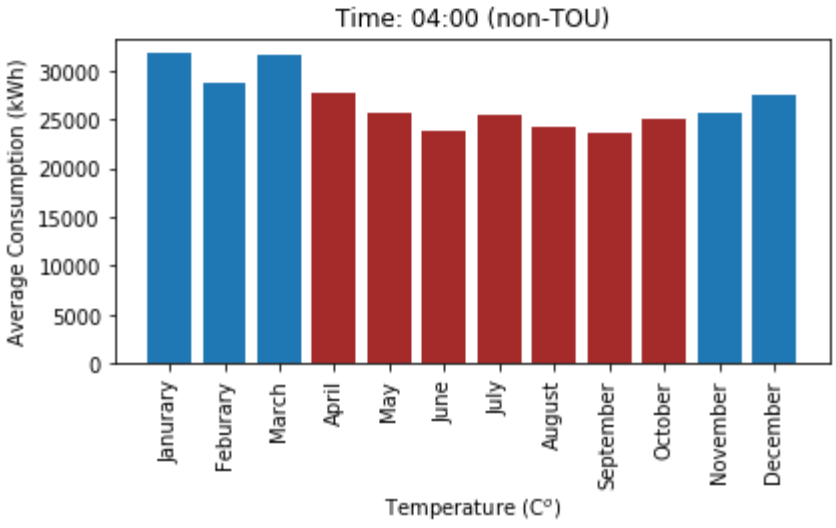

Time: 01:00 (non-TOU)
Average Consumption (kWh)
0
2000
4000
6000
8000
Janurary
Feburary
March
April
May
June
July
August
September
October
November
December
Temperature (C°)

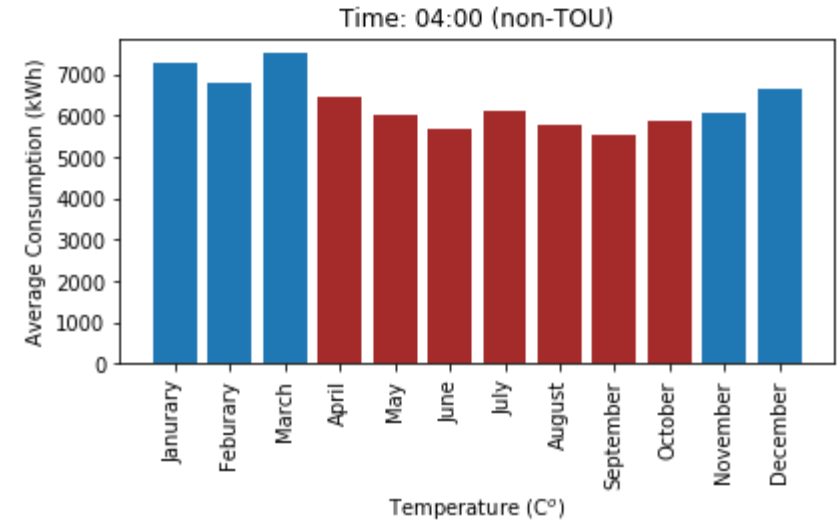

Time: 02:00 (non-TOU)
Average Consumption (kWh)
0
5000
10000
15000
20000
25000
30000
35000
Janurary
Feburary
March
April
May
June
July
August
September
October
November
December
Temperature (C°)


Time: 02:00 (non-TOU)
Average Consumption (kWh)
0
2000
4000
6000
8000
Janurary
Feburary
March
April
May
June
July
August
September
October
November
December
Temperature (C°)


Time: 03:00 (non-TOU)
Average Consumption (kWh)
0
5000
10000
15000
20000
25000
30000
Janurary
Feburary
March
April
May
June
July
August
September
October
November
December
Temperature (C°)

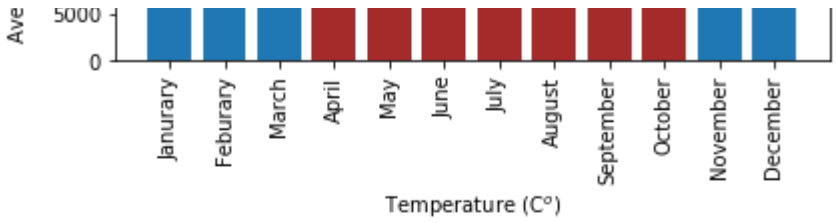

Time: 03:00 (non-TOU)
Average Consumption (kWh)
0
2000
4000
6000
8000
Janurary
Feburary
March
April
May
June
July
August
September
October
November
December
Temperature (C°)

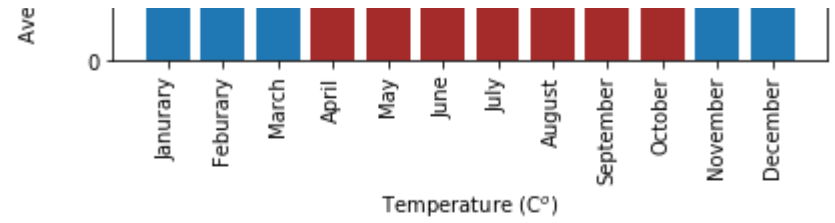

Time: 04:00 (non-TOU)
Average Consumption (kWh)
0
5000
10000
15000
20000
25000
30000
Janurary
Feburary
March
April
May
June
July
August
September
October
November
December
Temperature (C°)

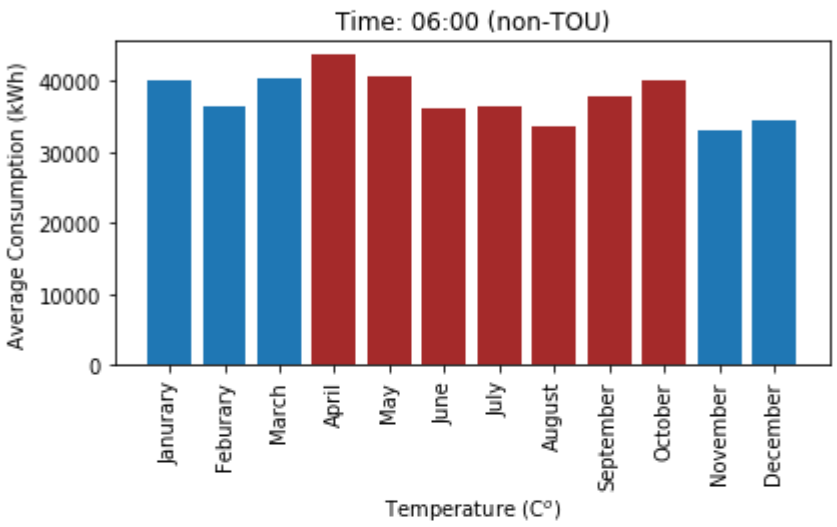

Time: 04:00 (non-TOU)
Average Consumption (kWh)
0
1000
2000
3000
4000
5000
6000
7000
Janurary
Feburary
March
April
May
June
July
August
September
October
November
December
Temperature (C°)

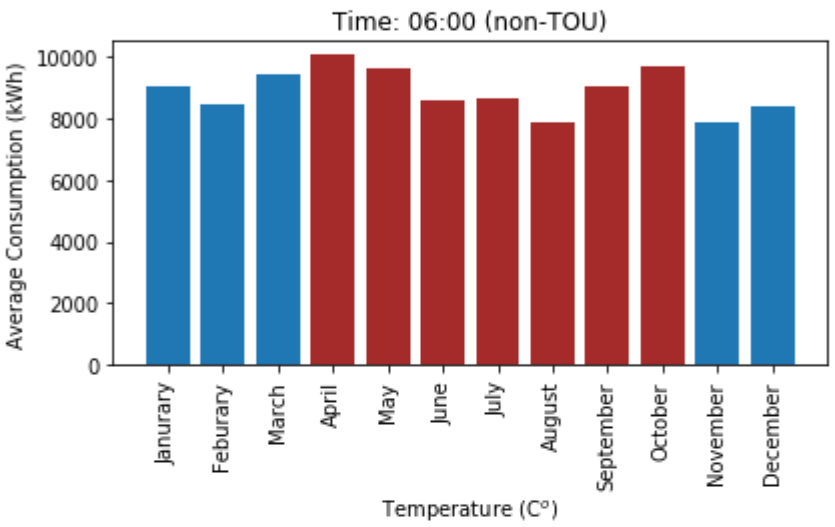
Time: 05:00 (non-TOU)
Average Consumption (kWh)
0
5000
10000
15000
20000
25000
30000
35000

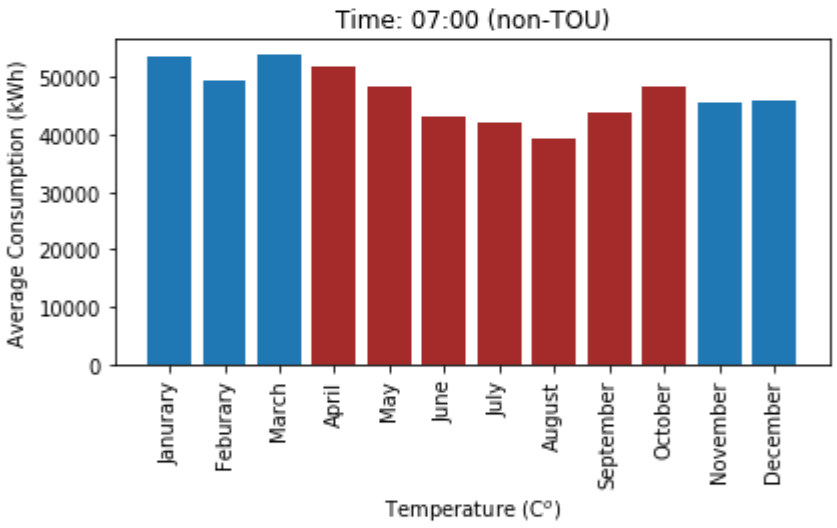
Time: 05:00 (non-TOU)
Average Consumption (kWh)
2000
4000
6000
8000

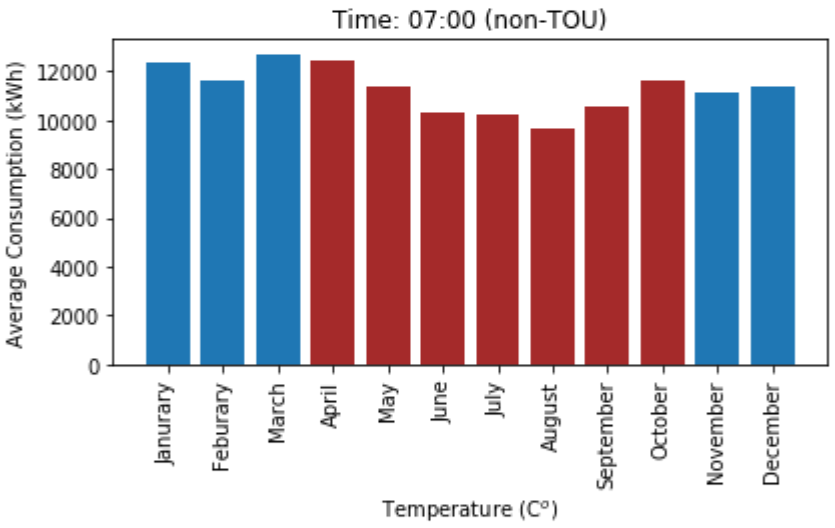

Ave
5000
0
Janurary
Feburary
March
April
May
June
July
August
September
October
November
December
Temperature (C°)

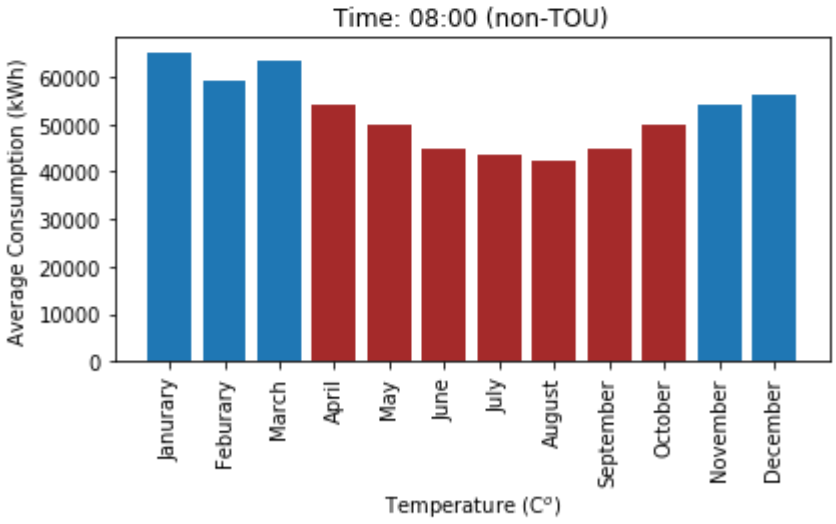

Ave
0
Janurary
Feburary
March
April
May
June
July
August
September
October
November
December
Temperature (C°)

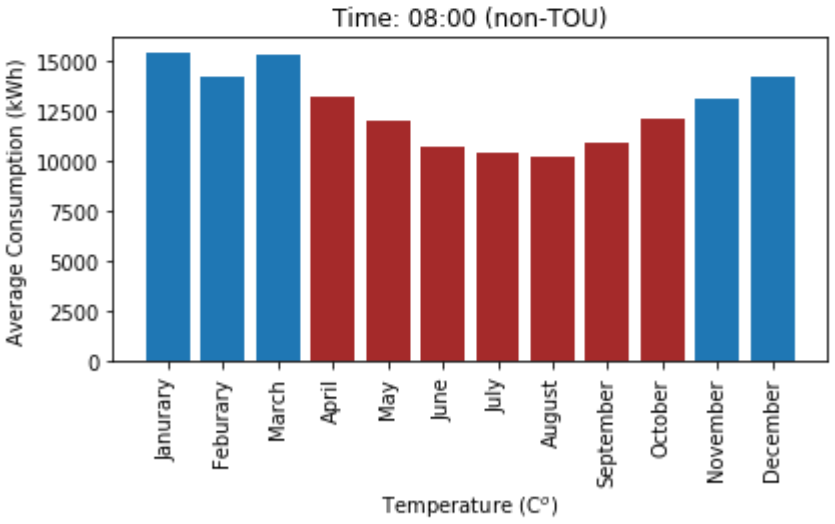

Time: 06:00 (non-TOU)
Average Consumption (kWh)
40000
30000
20000
10000
0
Janurary
Feburary
March
April
May
June
July
August
September
October
November
December
Temperature (C°)

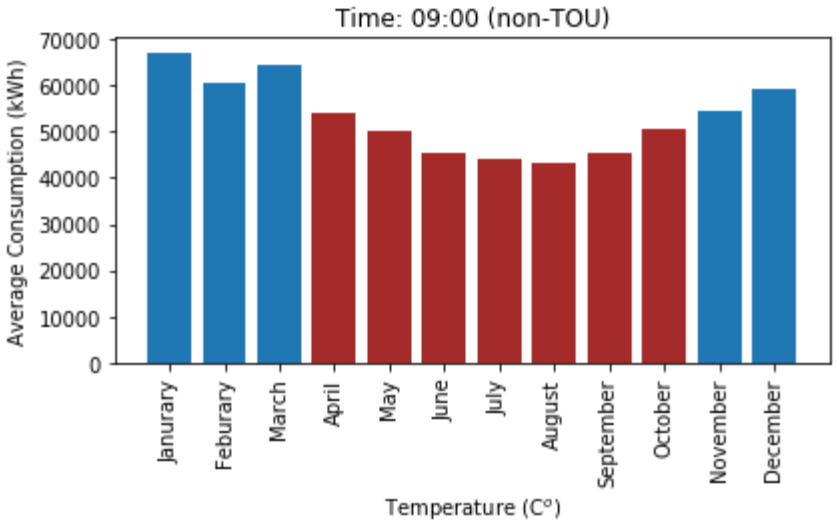

Time: 06:00 (non-TOU)
Average Consumption (kWh)
10000
8000
6000
4000
2000
0
Janurary
Feburary
March
April
May
June
July
August
September
October
November
December
Temperature (C°)

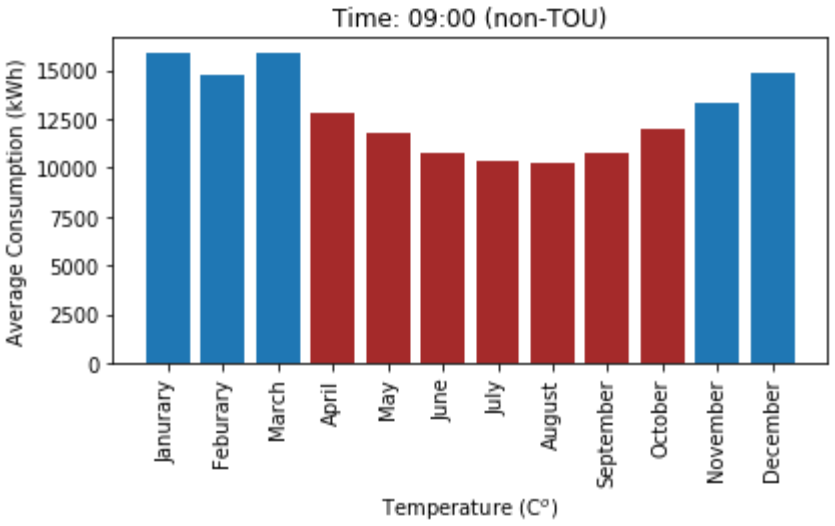

Time: 07:00 (non-TOU)
Average Consumption (kWh)
50000
40000
30000
20000
10000
0
Janurary
Feburary
March
April
May
June
July
August
September
October
November
December
Temperature (C°)

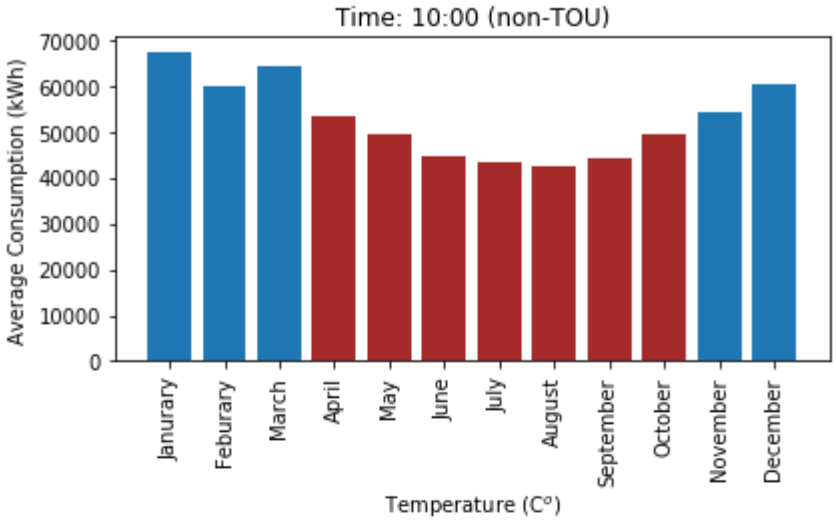

Time: 07:00 (non-TOU)
Average Consumption (kWh)
12000
10000
8000
6000
4000
2000
0
Janurary
Feburary
March
April
May
June
July
August
September
October
November
December
Temperature (C°)

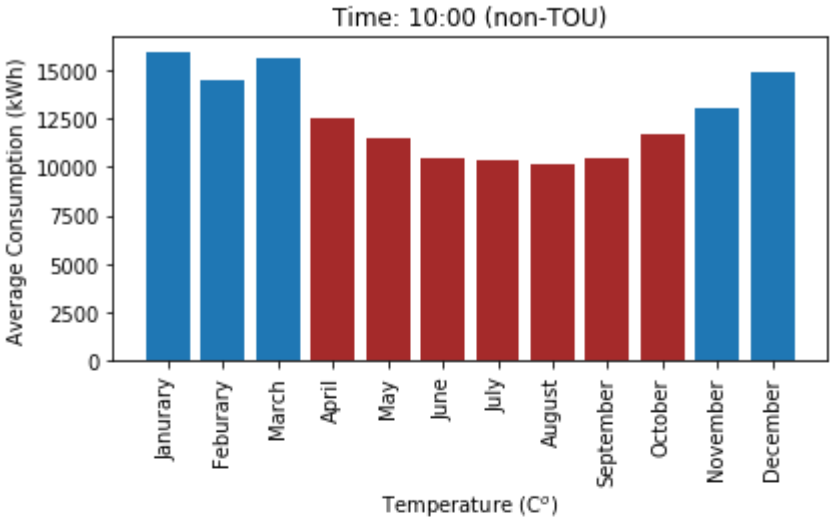

Time: 08:00 (non-TOU)
Average Consumption (kWh)
60000
50000
40000
30000
20000
10000
0
Janurary
Feburary
March
April
May
June
July
August
September
October
November
December
Temperature (C°)


Time: 08:00 (non-TOU)
Average Consumption (kWh)
15000
12500
10000
7500
5000
2500
0
Janurary
Feburary
March
April
May
June
July
August
September
October
November
December
Temperature (C°)


Time: 09:00 (non-TOU)
Average Consumption (kWh)
70000
60000
50000
40000
30000
20000
10000
0
Janurary
Feburary
March
April
May
June
July
August
September
October
November
December
Temperature (C°)

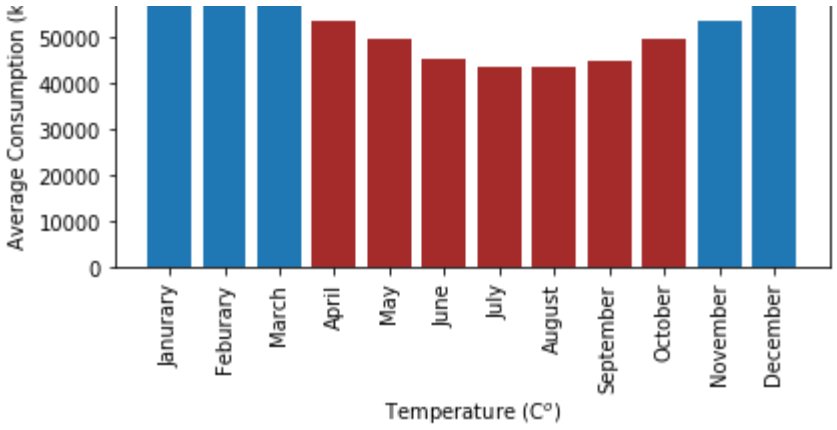

Time: 09:00 (non-TOU)
Average Consumption (kWh)
15000
12500
10000
7500
5000
2500
0
Janurary
Feburary
March
April
May
June
July
August
September
October
November
December
Temperature (C°)

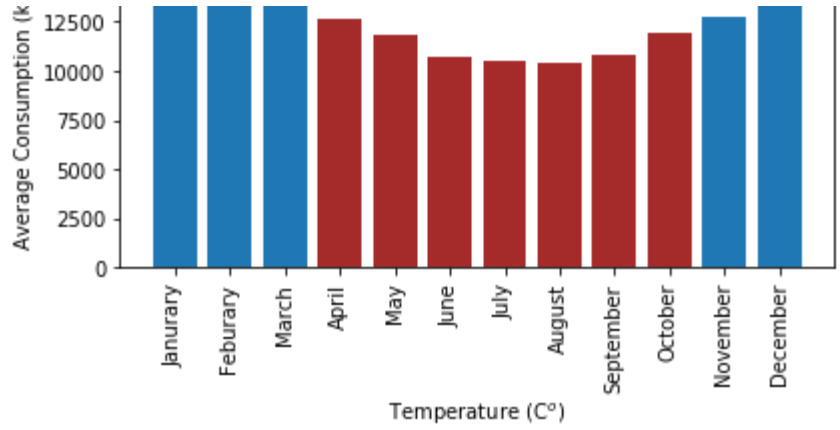

Time: 10:00 (non-TOU)
Average Consumption (kWh)
70000
60000
50000
40000
30000
20000
10000
0
Janurary
Feburary
March
April
May
June
July
August
September
October
November
December
Temperature (C°)

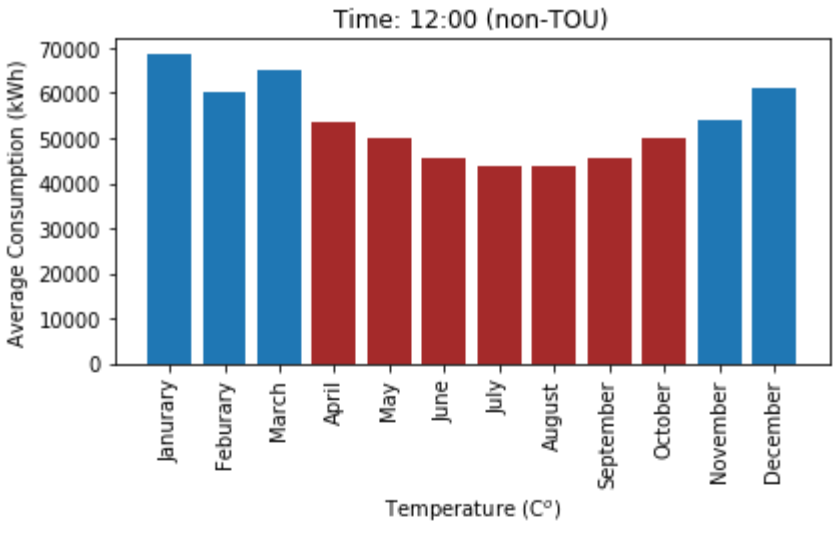

Time: 10:00 (non-TOU)
Average Consumption (kWh)
15000
12500
10000
7500
5000
2500
0
Janurary
Feburary
March
April
May
June
July
August
September
October
November
December
Temperature (C°)

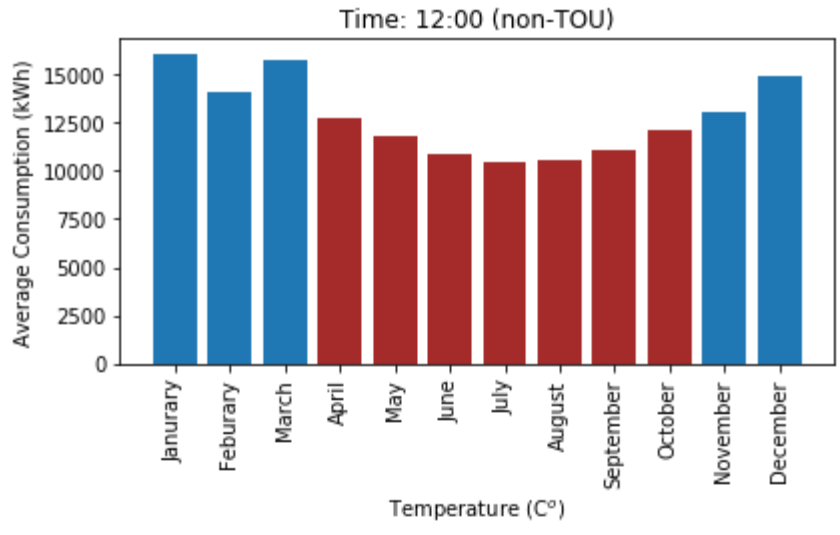
Time: 11:00 (non-TOU)
Average Consumption (kWh)
70000
60000

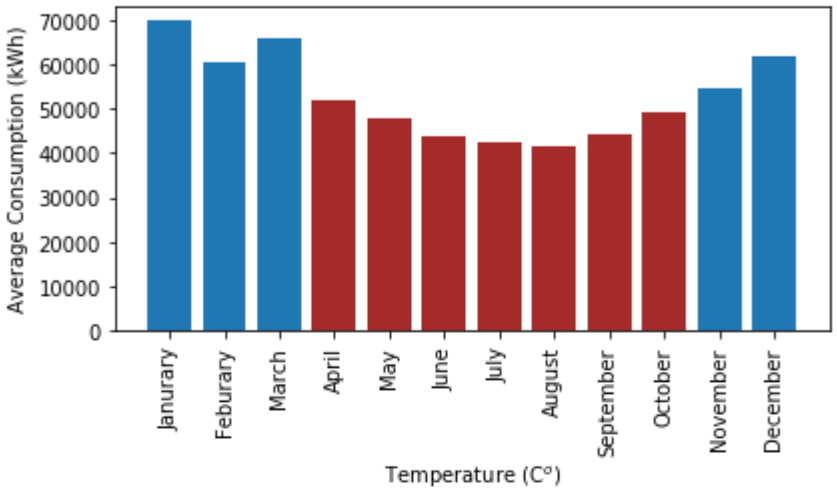
Time: 11:00 (non-TOU)
Average Consumption (kWh)
15000

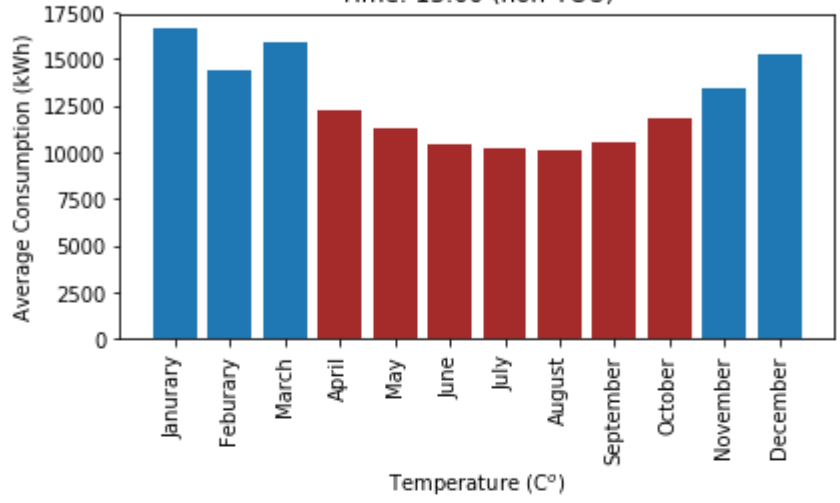

Average Consumption (k
0
10000
20000
30000
40000
50000
Janurary
Feburary
March
April
May
June
July
August
September
October
November
December
Temperature (C°)

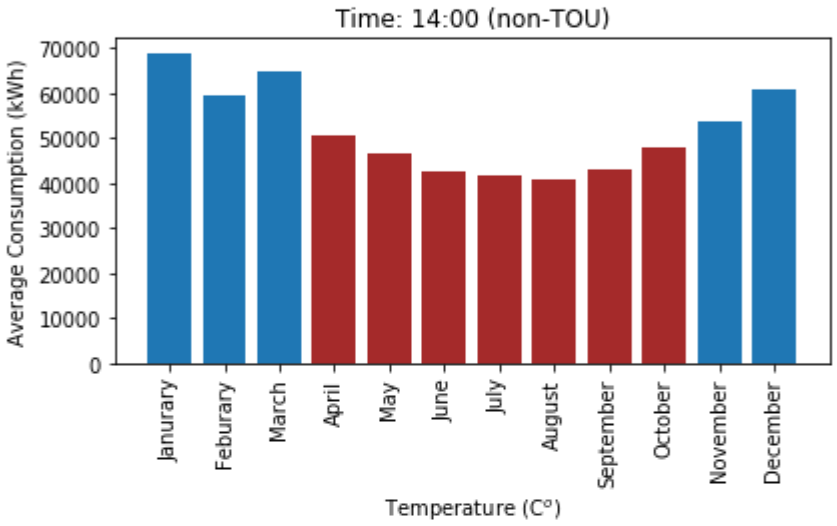

Average Consumption (k
0
2500
5000
7500
10000
12500
Janurary
Feburary
March
April
May
June
July
August
September
October
November
December
Temperature (C°)

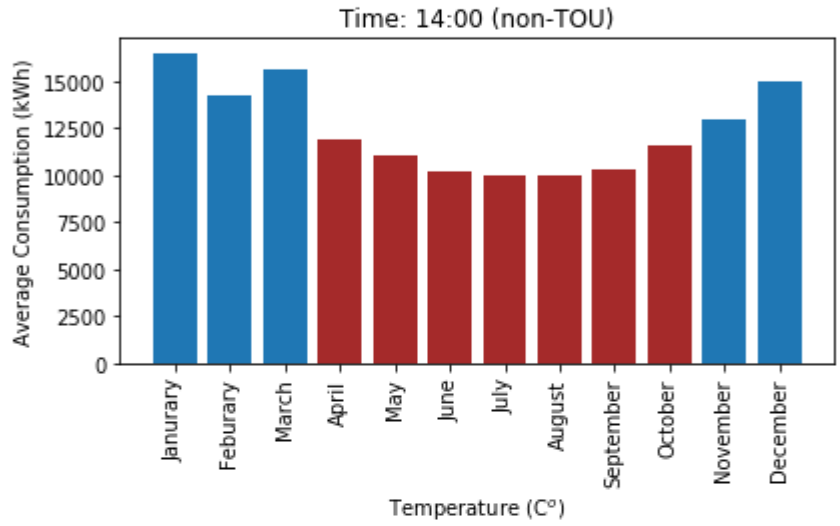

Time: 12:00 (non-TOU)
Average Consumption (kWh)
0
10000
20000
30000
40000
50000
60000
70000
Janurary
Feburary
March
April
May
June
July
August
September
October
November
December
Temperature (C°)

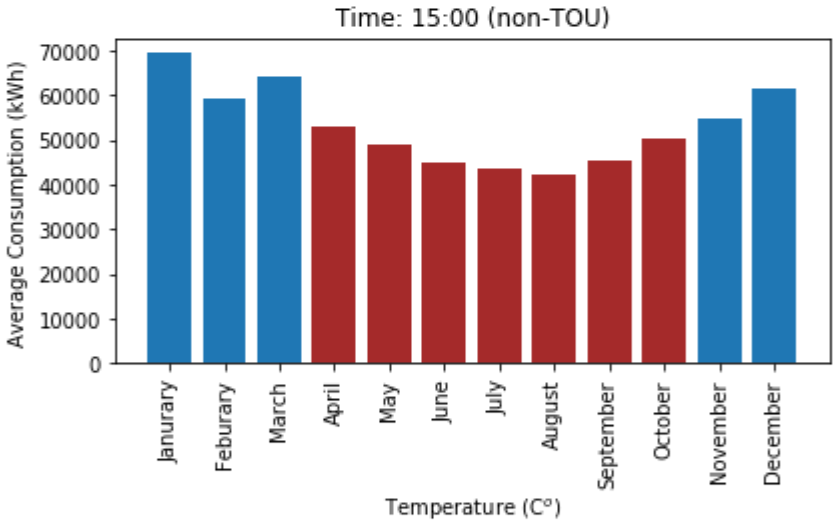

Time: 12:00 (non-TOU)
Average Consumption (kWh)
0
2500
5000
7500
10000
12500
15000
Janurary
Feburary
March
April
May
June
July
August
September
October
November
December
Temperature (C°)

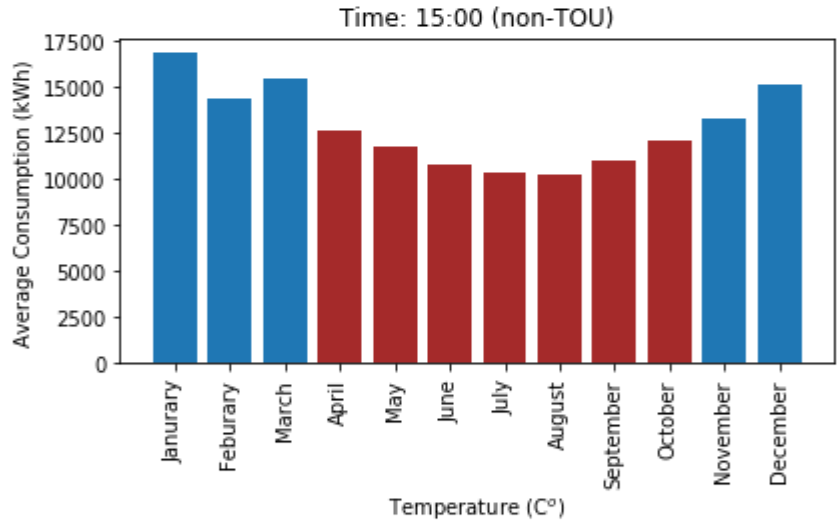

Time: 13:00 (non-TOU)
Average Consumption (kWh)
0
10000
20000
30000
40000
50000
60000
70000
Janurary
Feburary
March
April
May
June
July
August
September
October
November
December
Temperature (C°)

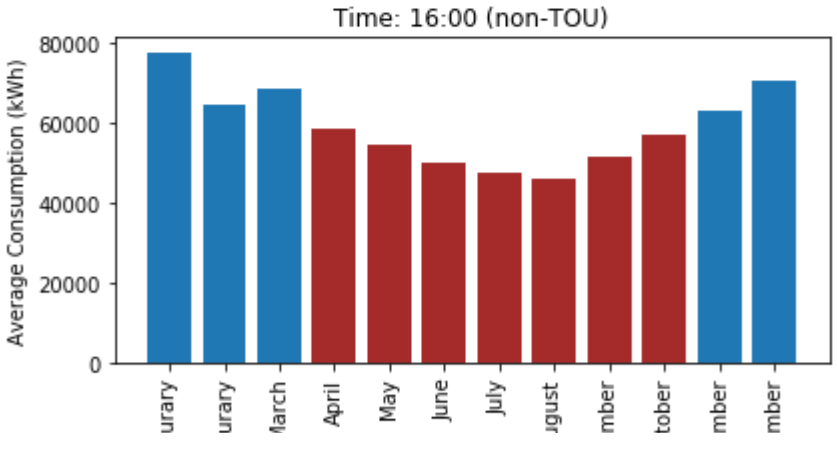

Time: 13:00 (non-TOU)
Average Consumption (kWh)
0
2500
5000
7500
10000
12500
15000
17500
Janurary
Feburary
March
April
May
June
July
August
September
October
November
December
Temperature (C°)

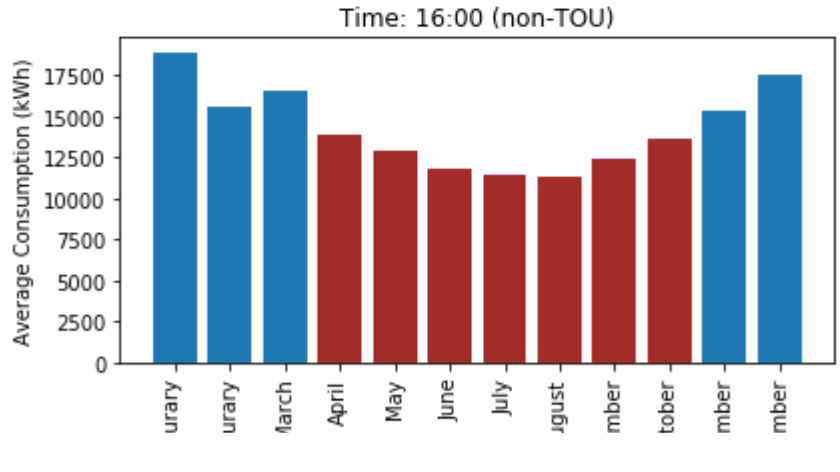

Time: 14:00 (non-TOU)
Average Consumption (kWh)
0
10000
20000
30000
40000
50000
60000
70000
Janurary
Feburary
March
April
May
June
July
August
September
October
November
December
Temperature (C°)

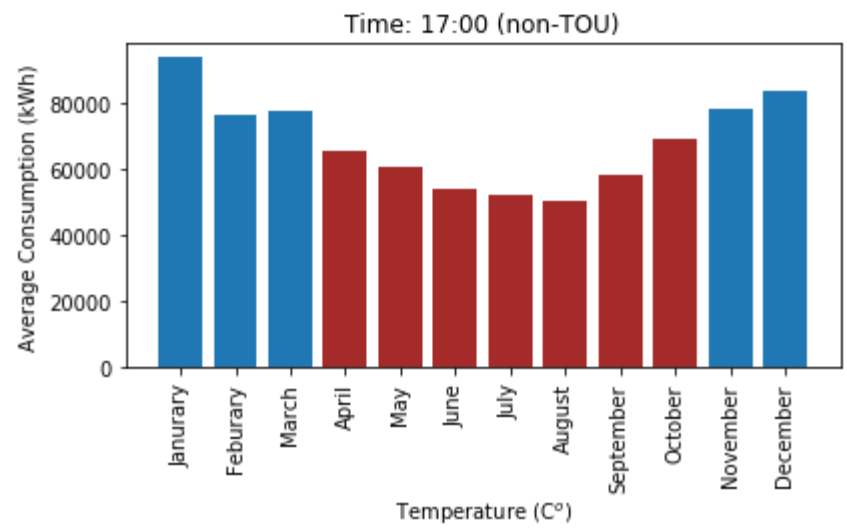

Time: 14:00 (non-TOU)
Average Consumption (kWh)
0
2500
5000
7500
10000
12500
15000
Janurary
Feburary
March
April
May
June
July
August
September
October
November
December
Temperature (C°)

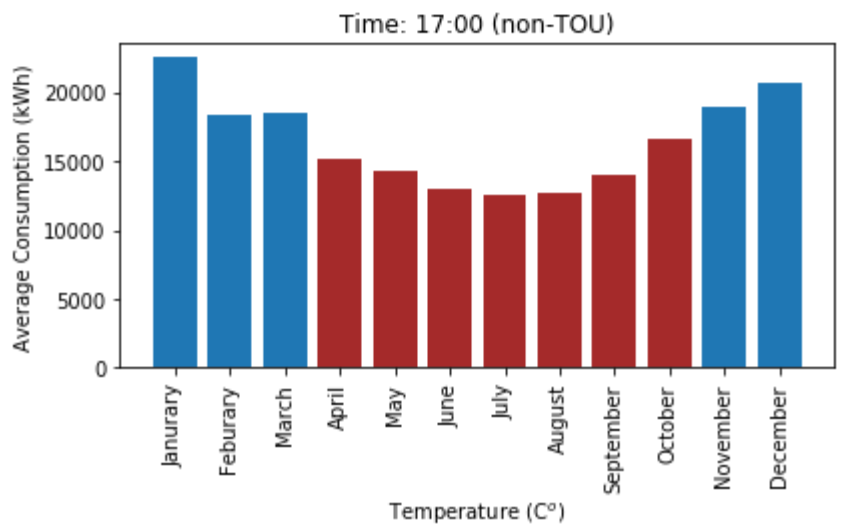

Time: 15:00 (non-TOU)
Average Consumption (kWh)
0
10000
20000
30000
40000
50000
60000
70000
Janurary
Feburary
March
April
May
June
July
August
September
October
November
December
Temperature (C°)

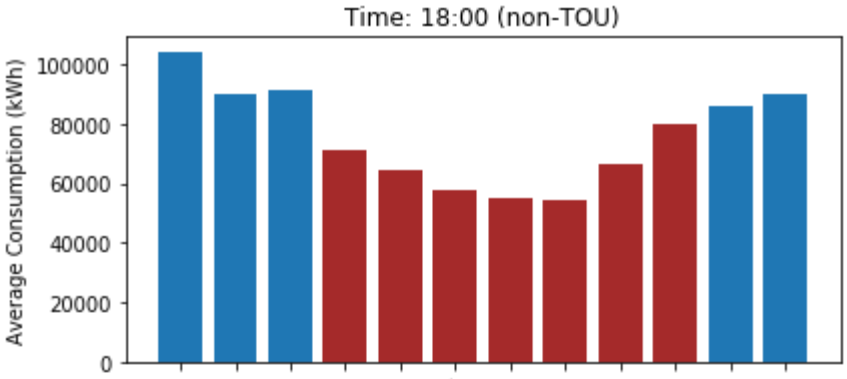

Time: 15:00 (non-TOU)
Average Consumption (kWh)
0
2500
5000
7500
10000
12500
15000
17500
Janurary
Feburary
March
April
May
June
July
August
September
October
November
December
Temperature (C°)

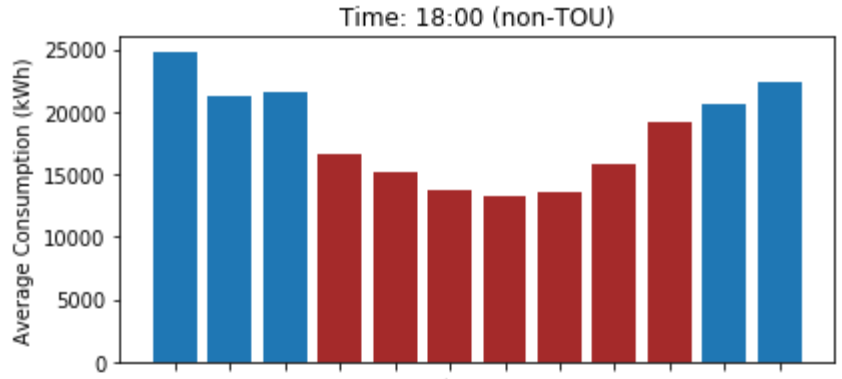

Time: 16:00 (non-TOU)
Average Consumption (kWh)
0
20000
40000
60000
80000
urary
urary
March
April
May
June
July
ugust
mber
tober
mber
mber

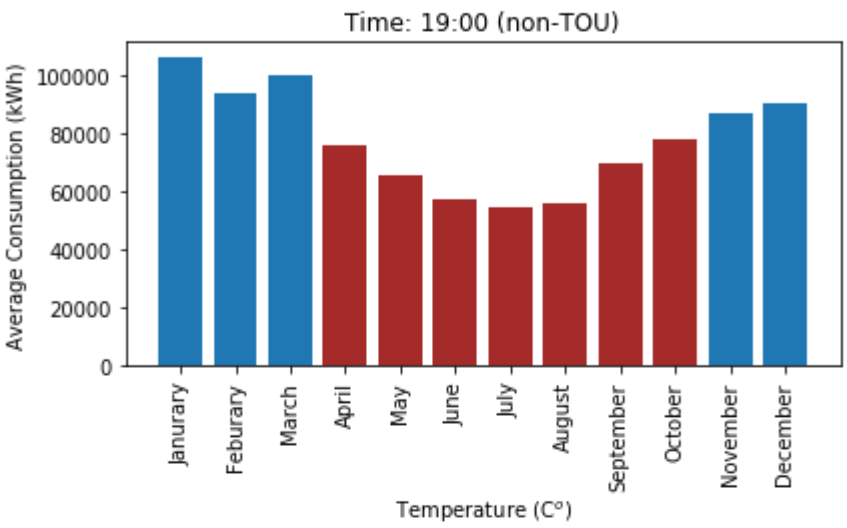

Time: 16:00 (non-TOU)
Average Consumption (kWh)
0
2500
5000
7500
10000
12500
15000
17500
urary
urary
March
April
May
June
July
ugust
mber
tober
mber
mber

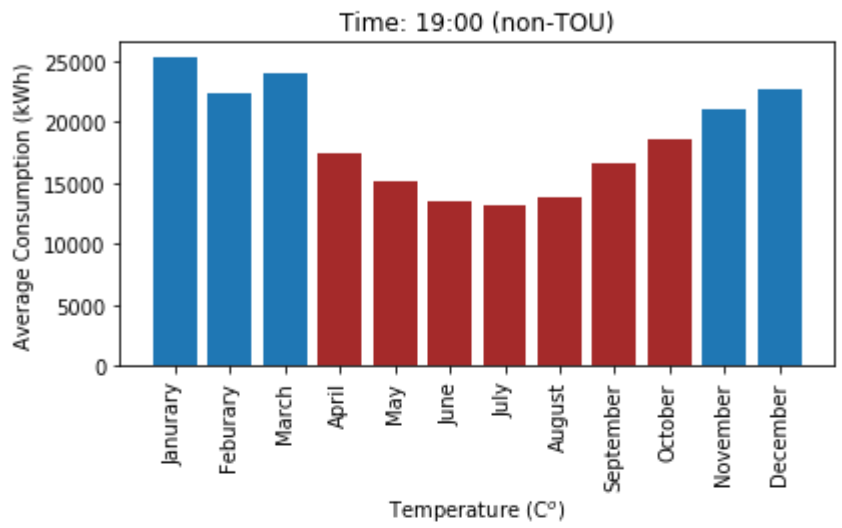

Time: 17:00 (non-TOU)
Average Consumption (kWh)
0
20000
40000
60000
80000
Janurary
Feburary
March
April
May
June
July
August
September
October
November
December
Temperature (C°)

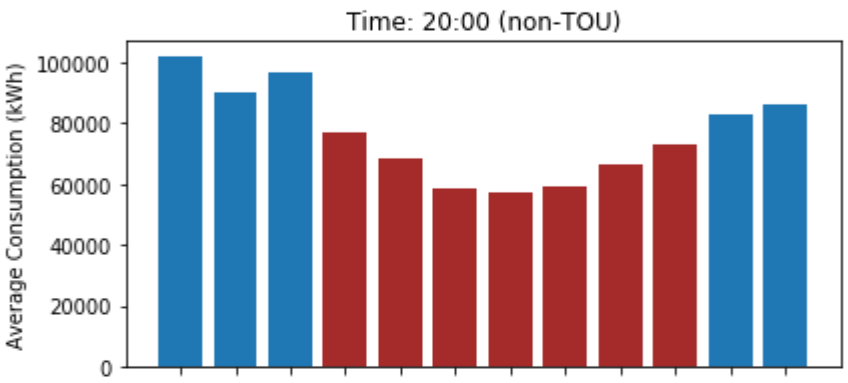

Time: 17:00 (non-TOU)
Average Consumption (kWh)
0
5000
10000
15000
20000
Janurary
Feburary
March
April
May
June
July
August
September
October
November
December
Temperature (C°)

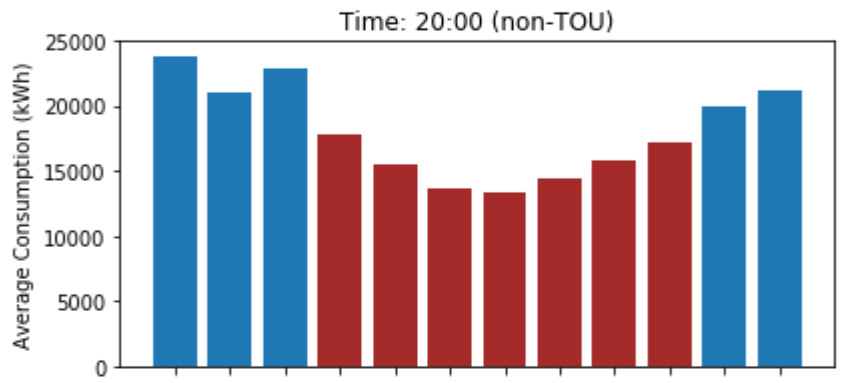

Time: 18:00 (non-TOU)
Average Consumption (kWh)
0
20000
40000
60000
80000
100000
Janurary
Feburary
March
April
May
June
July
August
September
October
November
December
Temperature (C°)

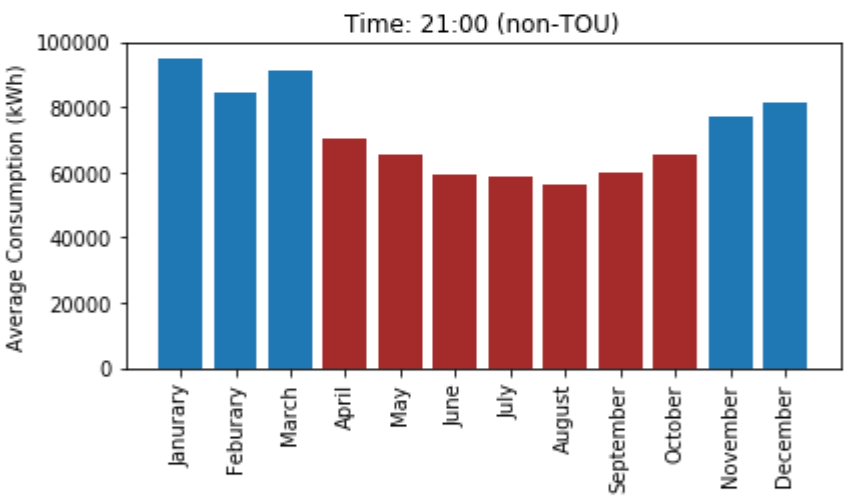

Time: 18:00 (non-TOU)
Average Consumption (kWh)
0
5000
10000
15000
20000
25000
Janurary
Feburary
March
April
May
June
July
August
September
October
November
December
Temperature (C°)

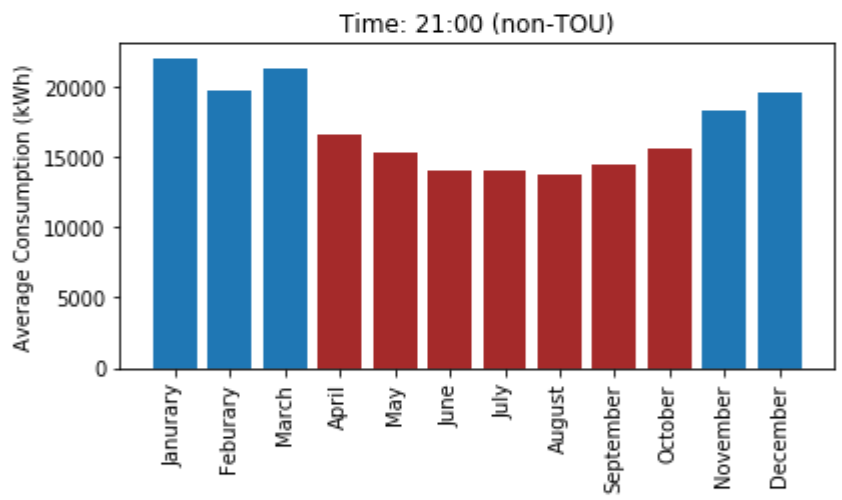

Time: 19:00 (non-TOU)
Average Consumption (kWh)
0
20000
40000
60000
80000
100000
Janurary
Feburary
March
April
May
June
July
August
September
October
November
December
Temperature (C°)

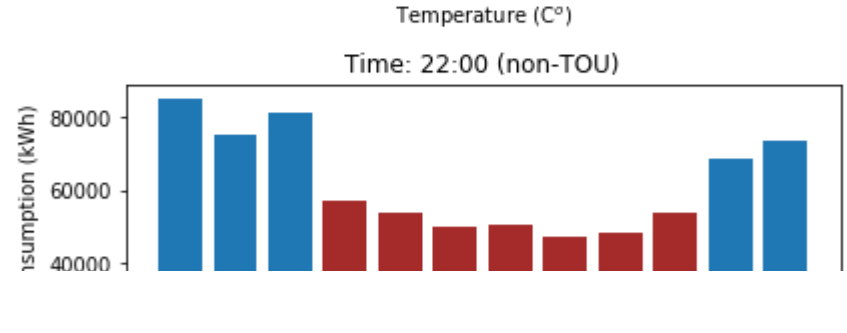

Time: 19:00 (non-TOU)
Average Consumption (kWh)
0
5000
10000
15000
20000
25000
Janurary
Feburary
March
April
May
June
July
August
September
October
November
December
Temperature (C°)

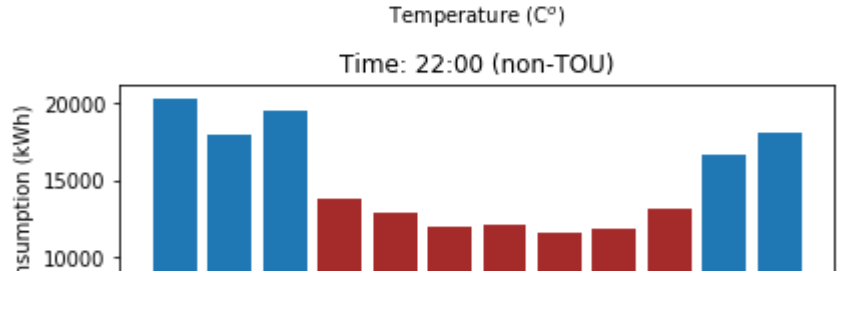

Time: 20:00 (non-TOU)
Average Consumption (kWh)
0
20000
40000
60000
80000
100000
Janurary
Feburary
March
April
May
June
July
August
September
October
November
December
Temperature (C°)

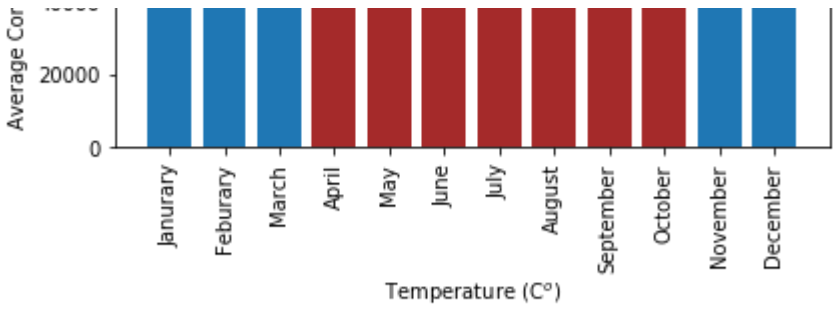

Time: 20:00 (non-TOU)
Average Consumption (kWh)
0
5000
10000
15000
20000
25000
Janurary
Feburary
March
April
May
June
July
August
September
October
November
December
Temperature (C°)

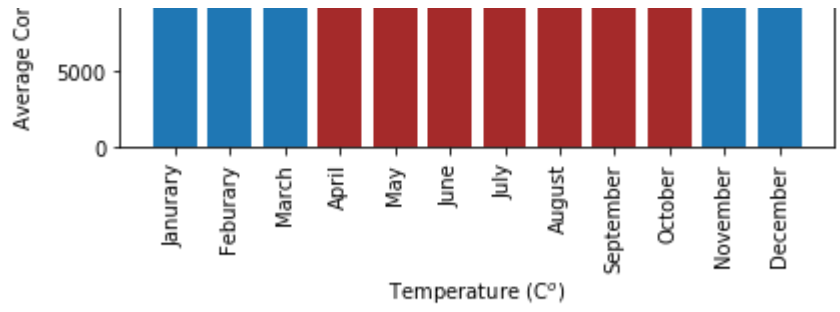

Time: 21:00 (non-TOU)
Average Consumption (kWh)
0
20000
40000
60000
80000
100000
Janurary
Feburary
March
April
May
June
July
August
September
October
November
December
Temperature (C°)

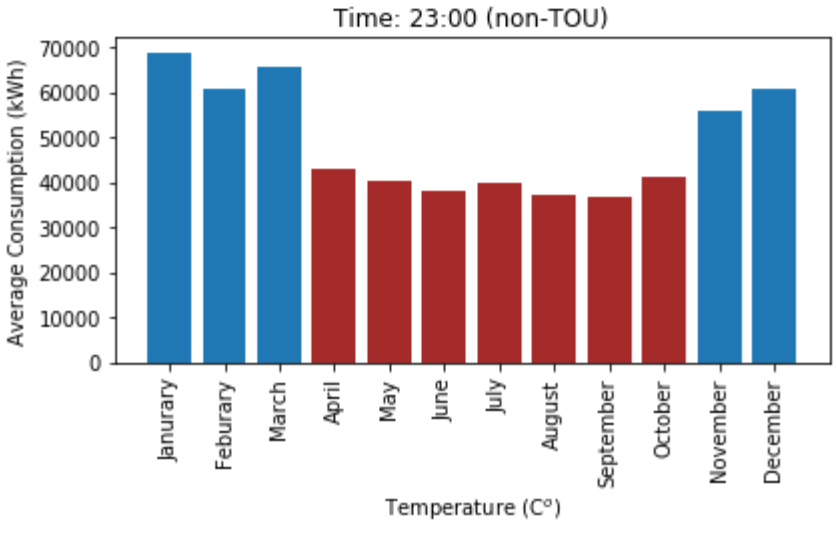

Time: 21:00 (non-TOU)
Average Consumption (kWh)
0
5000
10000
15000
20000
Janurary
Feburary
March
April
May
June
July
August
September
October
November
December
Temperature (C°)

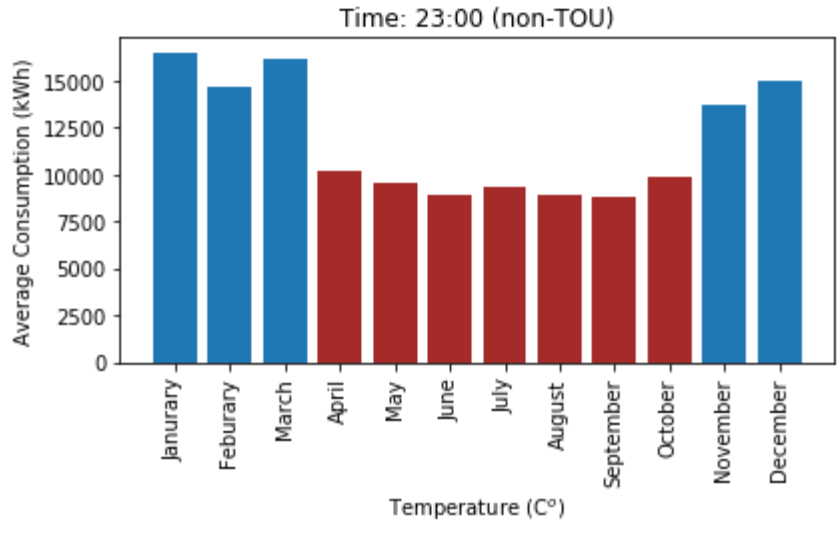

Time: 22:00 (non-TOU)
Average Consumption (kWh)
40000
60000
80000


Time: 22:00 (non-TOU)
Average Consumption (kWh)
10000
15000
20000


Average Cor
20000
0
Janurary
Feburary
March
April
May
June
July
August
September
October
November
December
Temperature (C°)


Average Cor
5000
0
Janurary
Feburary
March
April
May
June
July
August
September
October
November
December
Temperature (C°)


Time: 23:00 (non-TOU)
Average Consumption (kWh)
70000
60000
50000
40000
30000
20000
10000
0
Janurary
Feburary
March
April
May
June
July
August
September
October
November
December
Temperature (C°)


Time: 23:00 (non-TOU)
Average Consumption (kWh)
15000
12500
10000
7500
5000
2500
0
Janurary
Feburary
March
April
May
June
July
August
September
October
November
December
Temperature (C°)

In [14]:
```
# Create Spring, Summer, Autumn and Winter dataframe for load, price, weather
spring = [3, 4, 5]
autumn = [9, 10, 11]
df_Ntou1h_Day = df_Ntou1h[pd.to_datetime(df_tariff_1h.GMT).dt.weekday == 0]
# df_Ntou1h_Mon = df_Ntou1h[df_day_help]
# df_wea1h_spr3 = df_wea1h[df_month_help['GMT'] == 3]
# df_tou1h_aut11 = df_tou1h[df_month_help['GMT'] == 11]

# df_Ntou1h_aut11 = df_Ntou1h[df_month_help['GMT'] == 11]
df_Ntou1h_Day
```

Out[14]:

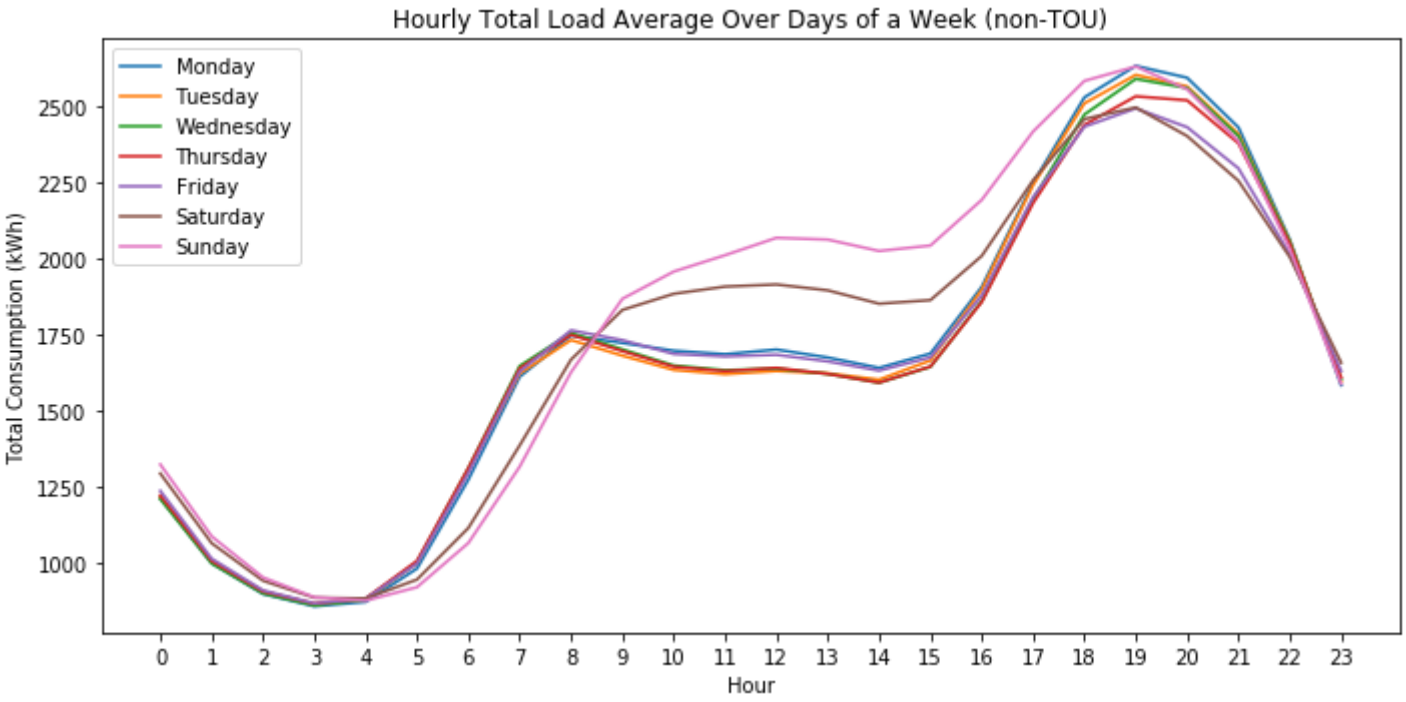
	GMT	N0000	N0001	N0002	N0003	N0004	N0005	N0006	N0007	N0008	...	N4166	N4167	N4168	N4169	N4170	N4171	N4
144	2013-01-07 00:00:00	0.375	0.259	0.079	0.108	0.375	0.153	0.000	0.156	0.131	...	0.062	0.326	0.218	0.124	0.147	0.112	0.1
145	2013-01-07 01:00:00	0.324	0.266	0.037	0.157	0.268	0.088	0.000	0.158	0.156	...	0.021	0.161	0.213	0.095	0.145	0.113	0.1
146	2013-01-07 02:00:00	0.317	0.229	0.057	0.133	0.274	0.078	0.000	0.164	0.134	...	0.061	0.187	0.171	0.082	0.158	0.112	0.1
147	2013-01-07 03:00:00	0.330	0.265	0.098	0.095	0.118	0.078	0.000	0.153	0.177	...	0.033	0.151	0.185	0.094	0.107	0.110	0.6
148	2013-01-07 04:00:00	0.416	0.267	0.088	0.069	0.119	0.087	0.000	0.169	0.121	...	0.021	0.171	0.169	0.080	0.116	0.111	0.3
149	2013-01-07 05:00:00	0.603	0.229	0.031	0.054	0.236	0.082	0.000	0.149	0.166	...	0.397	0.325	0.138	0.081	0.135	0.115	0.1
150	2013-01-07 06:00:00	0.480	0.265	0.047	0.106	0.142	0.534	0.000	0.170	0.133	...	2.088	1.455	0.178	0.100	0.091	0.137	0.1
151	2013-01-07 07:00:00	0.524	0.265	0.099	0.157	0.117	0.968	0.000	0.149	0.235	...	3.334	1.367	0.261	0.086	0.123	0.156	0.3
152	2013-01-07 08:00:00	0.347	0.225	0.472	0.260	0.114	0.591	0.000	0.167	0.483	...	1.970	0.475	0.400	0.085	0.128	0.135	0.0
153	2013-01-07 09:00:00	0.351	0.261	0.195	0.570	0.405	0.222	0.000	0.147	0.480	...	0.507	0.659	0.296	0.256	0.081	0.110	0.1
154	2013-01-07 10:00:00	0.347	0.259	0.352	0.489	0.561	0.219	0.000	0.164	0.407	...	0.021	1.893	0.125	0.144	0.349	0.175	0.1
155	2013-01-07 11:00:00	0.342	0.225	0.777	0.388	1.158	0.220	0.000	0.149	0.656	...	0.815	0.611	0.671	0.097	0.276	0.132	0.0
156	2013-01-07 12:00:00	0.378	0.264	0.426	0.080	0.630	0.212	0.000	0.160	1.095	...	0.466	0.672	0.403	0.078	0.527	0.112	0.3
157	2013-01-07 13:00:00	0.494	0.265	0.508	0.156	0.974	0.215	0.000	0.154	0.351	...	0.045	0.963	0.099	0.279	0.318	0.064	0.2
158	2013-01-07 14:00:00	0.320	0.225	0.353	0.193	0.449	0.223	0.000	0.155	0.313	...	0.045	0.780	0.976	0.079	0.383	0.591	0.3
159	2013-01-07 15:00:00	0.309	0.262	0.546	0.157	0.374	0.318	0.000	0.160	0.351	...	0.021	1.343	0.158	0.096	0.314	0.590	0.5
160	2013-01-07 16:00:00	1.071	0.260	0.594	0.286	0.476	0.258	0.000	0.148	0.373	...	0.134	2.378	0.285	0.084	0.238	0.241	0.5
161	2013-01-07 17:00:00	1.053	0.444	0.548	0.780	0.710	0.802	0.000	0.164	0.407	...	1.172	0.751	0.331	0.093	0.251	0.197	0.5
162	2013-01-07 18:00:00	0.433	1.212	0.517	0.710	0.786	1.916	0.000	0.187	0.555	...	2.676	1.750	0.320	0.246	0.396	0.135	1.0
163	2013-01-07 19:00:00	0.766	0.217	0.767	1.284	0.788	0.771	0.000	0.226	0.396	...	3.367	1.440	0.390	0.576	0.330	0.124	1.5
164	2013-01-07 20:00:00	0.593	0.083	0.593	0.951	0.843	0.813	0.000	0.194	2.066	...	3.772	0.658	0.349	0.523	0.537	0.125	0.6

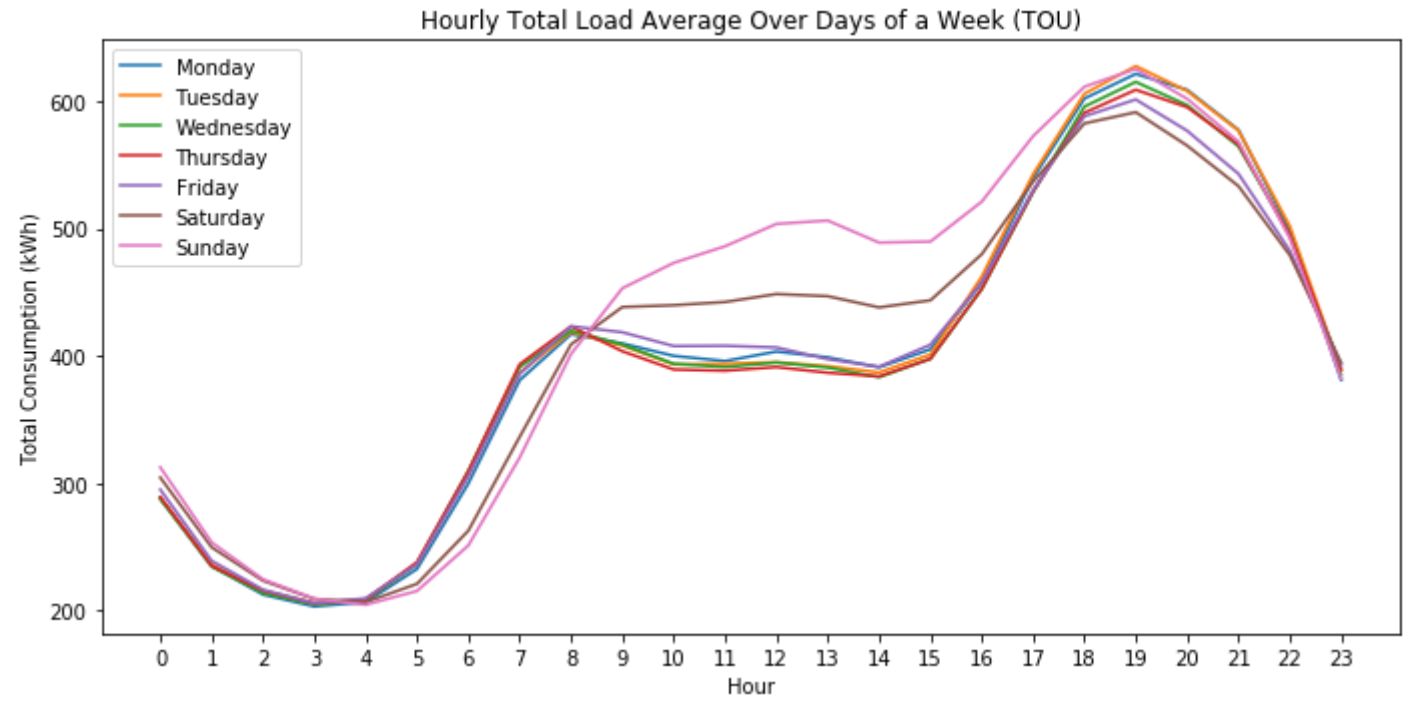
	GMT	N0000	N0001	N0002	N0003	N0004	N0005	N0006	N0007	N0008	...	N4166	N4167	N4168	N4169	N4170	N4171	N4
165	2013-01-07 21:00:00	0.420	0.116	0.481	1.043	0.593	0.719	0.000	0.225	0.551	...	2.391	3.315	0.335	0.485	0.287	0.118	1.4
166	2013-01-07 22:00:00	0.392	2.210	0.136	0.901	0.762	0.512	0.000	0.209	0.458	...	0.488	0.659	0.375	0.423	0.427	0.057	0.7
167	2013-01-07 23:00:00	0.353	1.098	0.070	0.221	0.230	0.373	0.000	0.194	0.331	...	0.259	0.255	0.271	0.298	0.271	0.021	0.5
312	2013-01-14 00:00:00	0.297	0.099	0.410	0.282	0.569	0.152	0.058	0.174	0.185	...	0.029	0.173	0.240	0.081	0.130	0.103	0.1
313	2013-01-14 01:00:00	0.332	0.099	0.130	0.201	1.358	0.157	0.000	0.158	0.150	...	0.021	0.105	0.222	0.083	0.146	0.086	0.1
314	2013-01-14 02:00:00	0.311	0.079	0.079	0.267	0.686	0.174	0.000	0.159	0.170	...	0.020	0.102	0.206	0.080	0.079	0.004	0.1
315	2013-01-14 03:00:00	0.306	0.085	0.119	0.206	0.117	0.146	0.047	0.158	0.153	...	0.028	0.109	0.336	0.096	0.151	0.009	0.4
316	2013-01-14 04:00:00	0.951	0.100	0.088	0.231	0.110	0.165	0.000	0.174	0.138	...	0.165	0.117	0.182	0.092	0.078	0.104	0.4
317	2013-01-14 05:00:00	1.110	0.114	0.095	0.173	0.166	0.142	0.000	0.163	0.181	...	0.400	0.071	0.194	0.080	0.079	0.096	0.2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
8562	2013-12-23 18:00:00	0.418	0.533	0.102	0.774	0.382	0.946	0.259	1.308	0.224	...	2.754	1.448	0.458	NaN	0.327	0.244	1.4
8563	2013-12-23 19:00:00	0.571	0.450	0.050	1.023	0.879	1.445	0.109	0.418	0.264	...	2.283	1.447	0.332	NaN	0.268	0.126	0.5
8564	2013-12-23 20:00:00	0.654	0.549	0.044	1.065	0.622	1.333	0.210	0.241	0.328	...	2.594	1.209	0.322	NaN	0.188	0.147	0.4
8565	2013-12-23 21:00:00	0.430	0.448	0.108	0.931	0.423	1.895	0.112	0.294	0.257	...	1.448	1.876	0.289	NaN	0.278	0.125	0.4
8566	2013-12-23 22:00:00	0.214	0.547	0.103	0.582	0.808	0.611	0.172	0.268	0.265	...	0.400	1.691	0.269	NaN	0.308	0.119	0.3
8567	2013-12-23 23:00:00	0.296	0.644	0.076	0.352	0.799	0.503	0.137	0.448	0.176	...	0.212	0.502	0.350	NaN	0.198	0.115	0.3
8712	2013-12-30 00:00:00	0.201	0.651	0.098	0.250	0.710	0.220	0.172	0.162	0.180	...	0.316	0.107	0.214	NaN	0.164	0.022	0.3
8713	2013-12-30 01:00:00	0.297	0.603	0.099	0.244	0.665	0.074	0.036	0.229	0.133	...	0.022	0.124	0.243	NaN	0.216	0.098	0.3
8714	2013-12-30 02:00:00	0.489	0.476	0.080	0.203	0.704	0.063	0.030	0.192	0.158	...	0.021	0.105	0.197	NaN	0.185	0.114	0.5
8715	2013-12-30 03:00:00	0.470	0.503	0.061	0.278	0.620	0.089	0.043	0.156	0.195	...	0.020	0.077	0.173	NaN	0.091	0.337	0.2
8716	2013-12-30 04:00:00	0.511	0.556	0.108	0.226	0.695	0.060	0.045	0.169	0.088	...	0.229	0.149	0.178	NaN	0.169	0.082	0.1
8717	2013-12-30 05:00:00	0.484	0.526	0.068	0.202	1.016	0.085	0.043	0.165	0.097	...	0.860	0.068	0.169	NaN	0.078	0.020	0.1

	GMT	N0000	N0001	N0002	N0003	N0004	N0005	N0006	N0007	N0008	...	N4166	N4167	N4168	N4169	N4170	N4171	N4
8718	2013-12-30 06:00:00	0.407	0.549	0.039	0.230	0.116	0.314	0.030	0.156	0.143	...	3.068	1.470	0.190	NaN	0.132	0.039	0.2
8719	2013-12-30 07:00:00	0.333	0.532	0.089	0.307	0.158	0.934	0.030	0.155	0.235	...	2.779	0.366	0.181	NaN	0.101	0.119	0.2
8720	2013-12-30 08:00:00	0.307	0.545	0.731	0.556	0.122	0.866	0.073	0.168	0.174	...	1.453	0.317	0.147	NaN	0.095	0.133	0.2
8721	2013-12-30 09:00:00	0.594	0.657	0.473	1.216	0.526	0.231	0.347	0.163	0.692	...	0.625	0.730	0.349	NaN	0.155	0.131	0.2
8722	2013-12-30 10:00:00	1.873	0.667	0.892	0.740	0.487	0.152	0.217	0.154	0.747	...	0.124	0.501	0.153	NaN	0.093	0.069	0.4
8723	2013-12-30 11:00:00	0.659	0.601	0.240	0.614	0.705	0.152	0.198	0.442	0.574	...	0.051	0.738	0.090	NaN	0.331	0.023	0.3
8724	2013-12-30 12:00:00	0.393	0.448	1.292	0.317	1.163	0.119	0.154	0.215	0.582	...	1.648	1.243	0.107	NaN	0.284	0.089	0.4
8725	2013-12-30 13:00:00	0.395	0.501	0.303	0.447	0.587	0.205	0.121	0.189	1.072	...	0.594	2.732	0.084	NaN	1.157	0.119	0.3
8726	2013-12-30 14:00:00	0.440	0.540	0.341	0.572	0.573	0.222	0.137	0.191	0.365	...	0.043	2.171	0.169	NaN	0.736	0.117	0.3
8727	2013-12-30 15:00:00	0.637	0.517	0.171	0.553	1.109	0.229	0.079	0.203	0.874	...	0.394	2.118	0.069	NaN	0.315	0.114	0.3
8728	2013-12-30 16:00:00	0.309	0.547	0.555	0.614	0.915	0.243	0.034	0.302	1.148	...	0.344	0.778	0.100	NaN	0.523	0.081	0.4
8729	2013-12-30 17:00:00	0.704	0.515	0.242	1.042	2.535	0.241	0.031	0.463	0.752	...	2.888	0.392	0.151	NaN	0.314	2.351	1.5
8730	2013-12-30 18:00:00	0.280	0.567	0.253	0.874	3.414	0.434	0.029	0.795	0.228	...	3.164	1.963	0.185	NaN	0.490	0.440	0.7
8731	2013-12-30 19:00:00	1.625	0.503	0.459	0.802	1.259	0.322	0.033	0.341	0.237	...	2.618	2.274	0.212	NaN	0.329	0.142	0.3
8732	2013-12-30 20:00:00	0.348	0.567	0.323	1.810	1.513	0.931	0.157	0.287	0.297	...	2.215	1.745	0.200	NaN	0.515	0.115	0.3
8733	2013-12-30 21:00:00	0.482	0.500	0.131	1.322	1.530	0.559	0.101	0.344	0.199	...	1.592	2.724	0.437	NaN	0.282	0.115	0.2
8734	2013-12-30 22:00:00	0.555	0.787	0.073	1.713	1.367	0.550	0.112	0.326	0.194	...	0.097	1.547	0.245	NaN	0.213	0.261	0.3
8735	2013-12-30 23:00:00	0.508	0.608	0.107	0.241	0.956	0.381	0.103	0.634	0.250	...	0.038	1.199	0.246	NaN	0.285	0.110	0.2

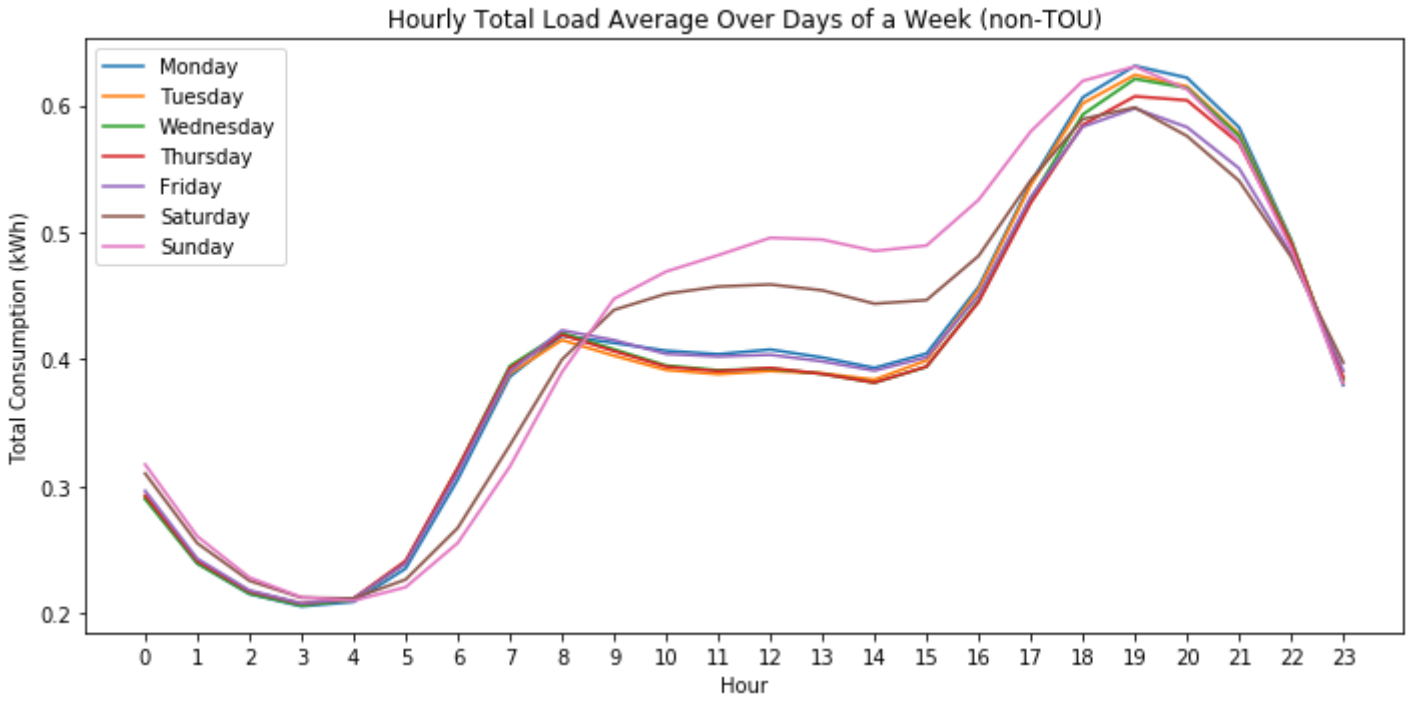
1248 rows × 4177 columns

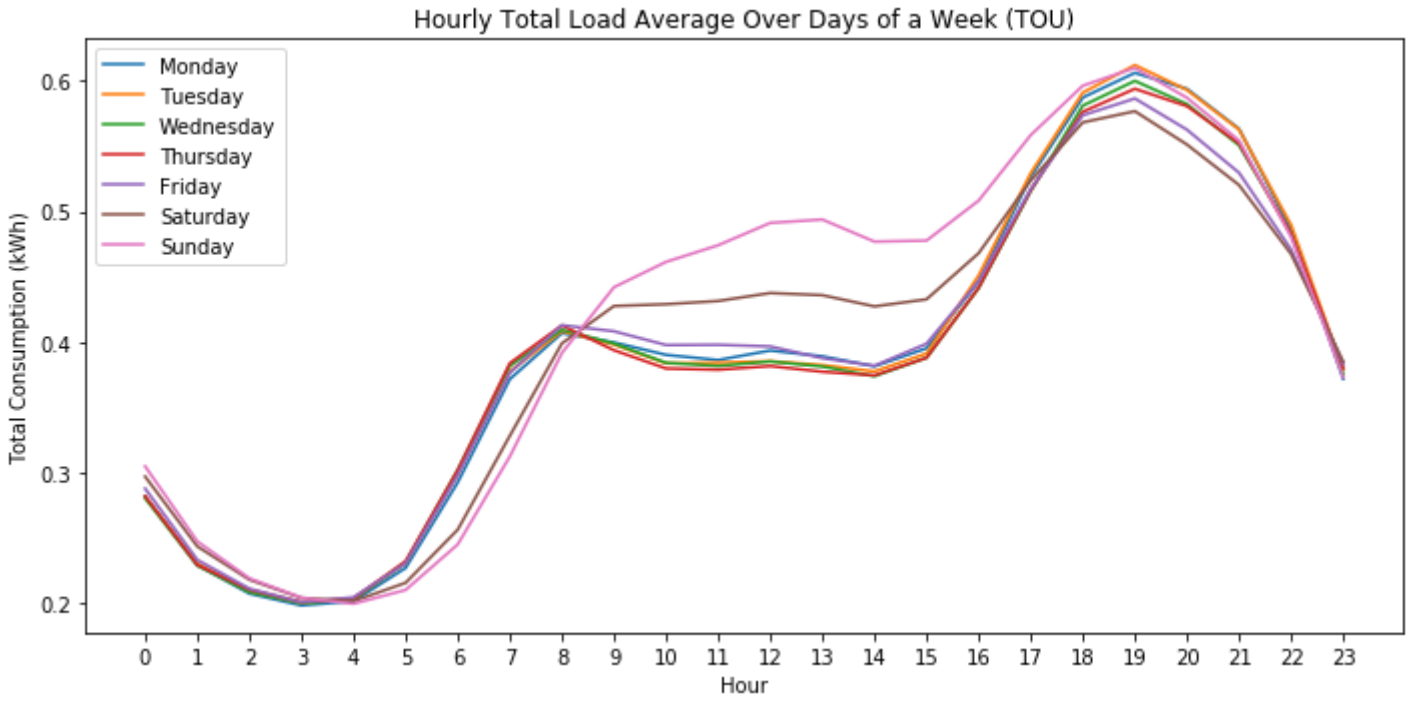
In [49]:
```
# Overview of the weekly patterns
# non-TOU vs TOU in terms of showing the elements in Average column such
# that they are hourly energy consumption over different days
# of a week for a whole year
week_days = [1, 2, 3, 4, 5, 6, 7]
label_week = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
# non-TOU vs TOU in terms of showing the elements in Average column such that they are hourly energy consumpt
ion v.s. temperature over 365 days
# fig_all = plt.figure(figsize = (12,90))
fig_all = plt.figure(figsize = (10, 10))
ax = [] # store subplot objects

ax.append(fig_all.add_subplot(2, 1, 1))
for j in range(len(week_days)):
    df_Ntou1h_Day = df_Ntou1h[pd.to_datetime(df_tariff_1h.GMT).dt.weekday == j]
    df_tou1h_Day = df_tou1h[pd.to_datetime(df_tariff_1h.GMT).dt.weekday == j]
    hour_totload_avg = []
    for i in range(24):
        if i <= 9:
            hour_totload_avg.append(df_Ntou1h_Day.Total[df_Ntou1h.GMT.str.contains('0' + str(i) + ':00:00')].
mean())
        else:
            hour_totload_avg.append(df_Ntou1h_Day.Total[df_Ntou1h.GMT.str.contains(str(i) + ':00:00')].mean
())
    ax[-1].plot(range(24), hour_totload_avg, label = label_week[j])
ax[-1].set_title('Hourly Total Load Average Over Days of a Week (non-TOU)')
ax[-1].set_xlabel('Hour')
ax[-1].set_xticks(range(24))
ax[-1].set_ylabel('Total Consumption (kWh)')
ax[-1].legend()

ax.append(fig_all.add_subplot(2, 1, 2))
for j in range(len(week_days)):
    df_tou1h_Day = df_tou1h[pd.to_datetime(df_tariff_1h.GMT).dt.weekday == j]
    hour_totload_avg = []
    for i in range(24):
        if i <= 9:
            hour_totload_avg.append(df_tou1h_Day.Total[df_tou1h.GMT.str.contains('0' + str(i) + ':00:00')].me
an())
        else:
            hour_totload_avg.append(df_tou1h_Day.Total[df_tou1h.GMT.str.contains(str(i) + ':00:00')].mean())
    ax[-1].plot(hour_totload_avg, label = label_week[j])
ax[-1].set_title('Hourly Total Load Average Over Days of a Week (TOU)')
ax[-1].set_xlabel('Hour')
ax[-1].set_xticks(range(24))
ax[-1].set_ylabel('Total Consumption (kWh)')
ax[-1].legend()
plt.tight_layout()
```


Hourly Total Load Average Over Days of a Week (non-TOU)
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
Total Consumption (kWh)
2500
2250
2000
1750
1500
1250
1000
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

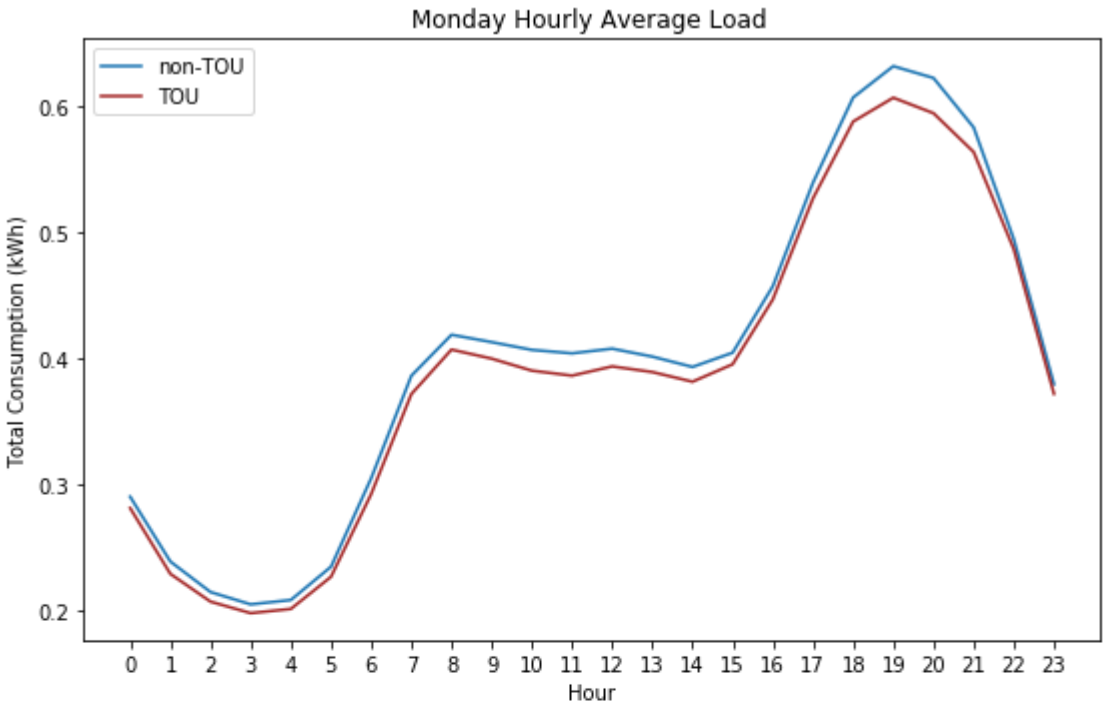

Hourly Total Load Average Over Days of a Week (TOU)
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
Total Consumption (kWh)
600
500
400
300
200
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

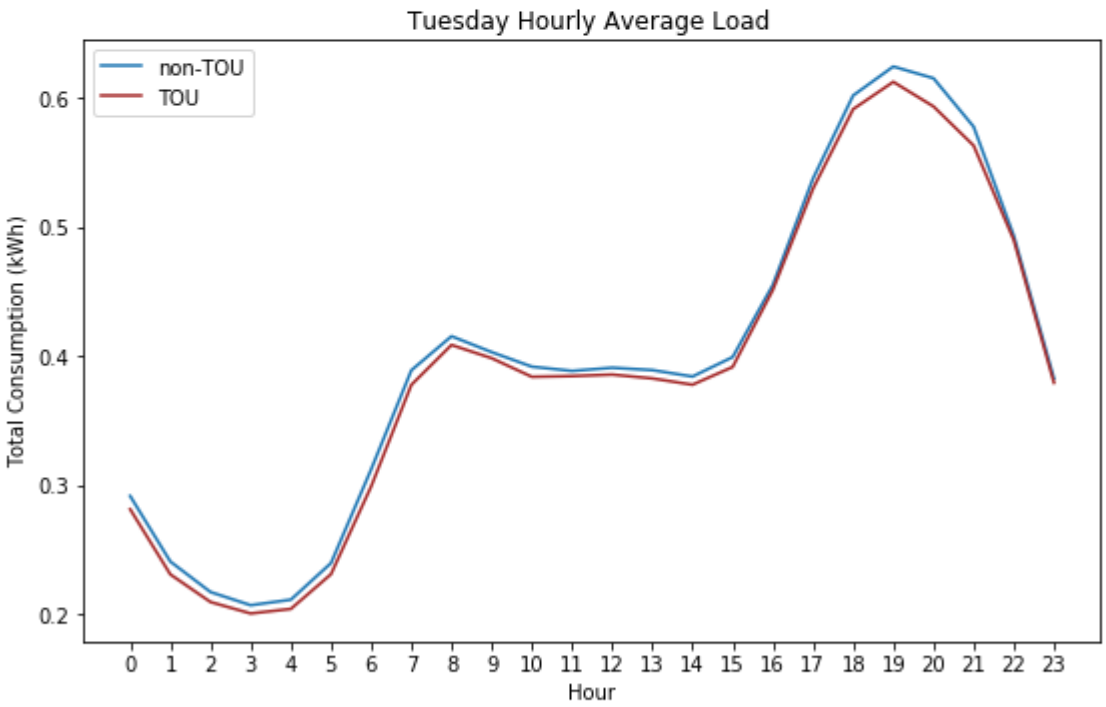
In [50]:
```
# Overview of the weekly patterns
# non-TOU vs TOU in terms of showing the elements in Average column such
# that they are hourly energy consumption over different days
# of a week for a whole year
week_days = [1, 2, 3, 4, 5, 6, 7]
label_week = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
# non-TOU vs TOU in terms of showing the elements in Average column such that they are hourly energy consumpt
ion v.s. temperature over 365 days
# fig_all = plt.figure(figsize = (12,90))
fig_all = plt.figure(figsize = (10, 10))
ax = [] # store subplot objects

ax.append(fig_all.add_subplot(2, 1, 1))
for j in range(len(week_days)):
    df_Ntou1h_Day = df_Ntou1h[pd.to_datetime(df_tariff_1h.GMT).dt.weekday == j]
    df_tou1h_Day = df_tou1h[pd.to_datetime(df_tariff_1h.GMT).dt.weekday == j]
    hour_totload_avg = []
    for i in range(24):
        if i <= 9:
            hour_totload_avg.append(df_Ntou1h_Day.Average[df_Ntou1h.GMT.str.contains('0' + str(i) + ':00:00'
)].mean())
        else:
            hour_totload_avg.append(df_Ntou1h_Day.Average[df_Ntou1h.GMT.str.contains(str(i) + ':00:00')].mean
())
    ax[-1].plot(range(24), hour_totload_avg, label = label_week[j])
ax[-1].set_title('Hourly Total Load Average Over Days of a Week (non-TOU)')
ax[-1].set_xlabel('Hour')
ax[-1].set_xticks(range(24))
ax[-1].set_ylabel('Total Consumption (kWh)')
ax[-1].legend()

ax.append(fig_all.add_subplot(2, 1, 2))
for j in range(len(week_days)):
    df_tou1h_Day = df_tou1h[pd.to_datetime(df_tariff_1h.GMT).dt.weekday == j]
    hour_totload_avg = []
    for i in range(24):
        if i <= 9:
            hour_totload_avg.append(df_tou1h_Day.Average[df_tou1h.GMT.str.contains('0' + str(i) + ':00:00')].
mean())
        else:
            hour_totload_avg.append(df_tou1h_Day.Average[df_tou1h.GMT.str.contains(str(i) + ':00:00')].mean
())
    ax[-1].plot(hour_totload_avg, label = label_week[j])
ax[-1].set_title('Hourly Total Load Average Over Days of a Week (TOU)')
ax[-1].set_xlabel('Hour')
ax[-1].set_xticks(range(24))
ax[-1].set_ylabel('Total Consumption (kWh)')
ax[-1].legend()
plt.tight_layout()
```

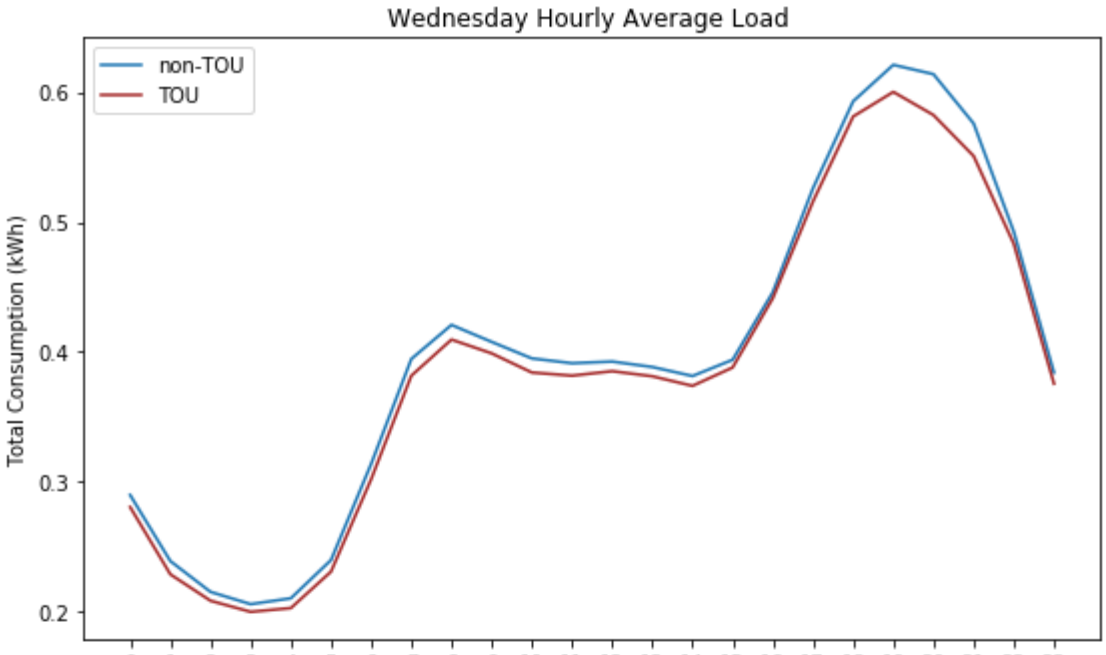

Hourly Total Load Average Over Days of a Week (non-TOU)
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
Total Consumption (kWh)
0.2
0.3
0.4
0.5
0.6
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

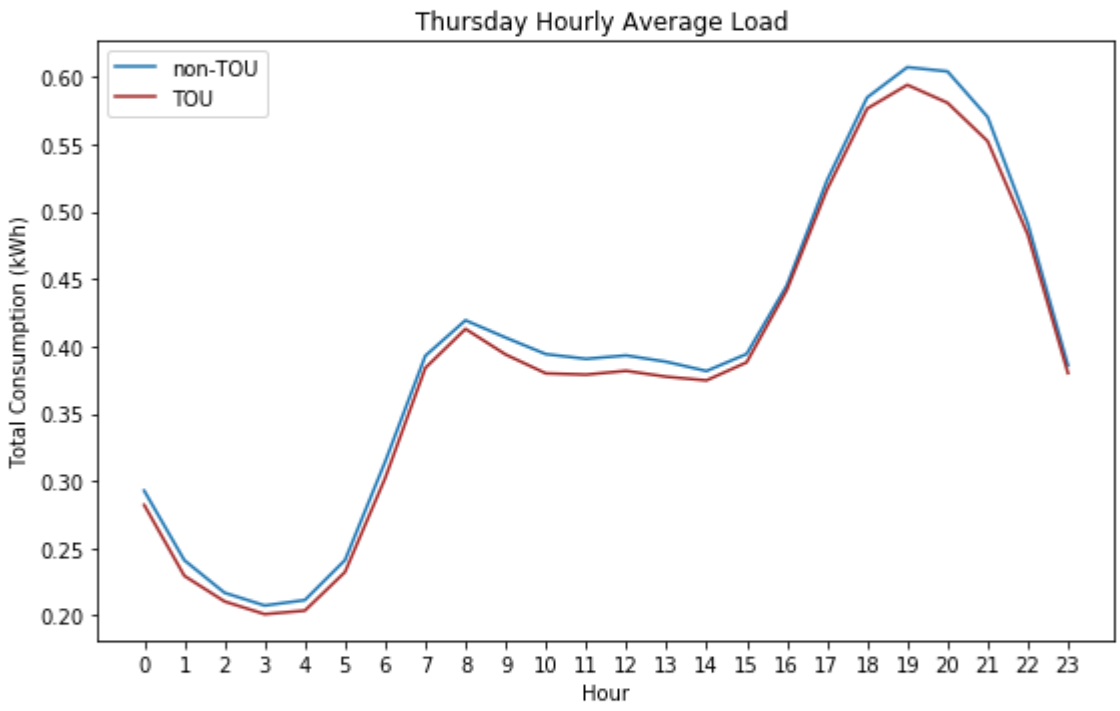

Hourly Total Load Average Over Days of a Week (TOU)
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
Consumption (kWh)
0.2
0.3
0.4
0.5
0.6
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

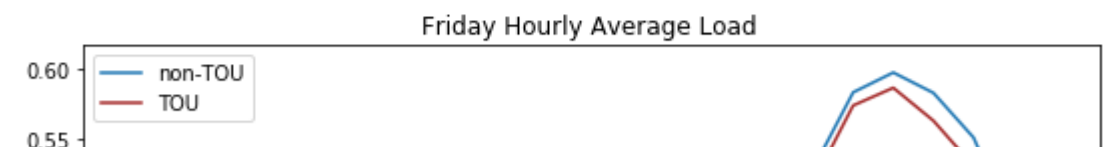
In [64]:
```
# Overview of the weekly patterns do it one day of TOU by one day of non-TOU
# non-TOU vs TOU in terms of showing the elements in Average column such
# that they are hourly energy consumption over different days
# of a week for a whole year
week_days = [1, 2, 3, 4, 5, 6, 7]
label_week = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
# non-TOU vs TOU in terms of showing the elements in Average column such that they are hourly energy consumpt
ion v.s. temperature over 365 days
# fig_all = plt.figure(figsize = (12,90))
fig_all = plt.figure(figsize = (8, 35))
ax = [] # store subplot objects

for j in range(len(week_days)):
    ax.append(fig_all.add_subplot(7, 1, j+1))
    df_Ntou1h_Day = df_Ntou1h[pd.to_datetime(df_tariff_1h.GMT).dt.weekday == j]
    df_tou1h_Day = df_tou1h[pd.to_datetime(df_tariff_1h.GMT).dt.weekday == j]
    hour_totload_avg = []
    for i in range(24):
        if i <= 9:
            hour_totload_avg.append(df_Ntou1h_Day.Average[df_Ntou1h.GMT.str.contains('0' + str(i) + ':00:00'
)].mean())
        else:
            hour_totload_avg.append(df_Ntou1h_Day.Average[df_Ntou1h.GMT.str.contains(str(i) + ':00:00')].mean
())
    ax[-1].plot(range(24), hour_totload_avg, label = 'non-TOU', color = 'C0')

    df_tou1h_Day = df_tou1h[pd.to_datetime(df_tariff_1h.GMT).dt.weekday == j]
    hour_totload_avg = []
    for i in range(24):
        if i <= 9:
            hour_totload_avg.append(df_tou1h_Day.Average[df_tou1h.GMT.str.contains('0' + str(i) + ':00:00')].
mean())
        else:
            hour_totload_avg.append(df_tou1h_Day.Average[df_tou1h.GMT.str.contains(str(i) + ':00:00')].mean
())
    ax[-1].plot(hour_totload_avg, label = 'TOU', color = 'brown')
    ax[-1].set_title(label_week[j] + ' Hourly Average Load ')
    ax[-1].set_xlabel('Hour')
    ax[-1].set_xticks(range(24))
    ax[-1].set_ylabel('Total Consumption (kWh)')
    ax[-1].legend()
plt.tight_layout()
```

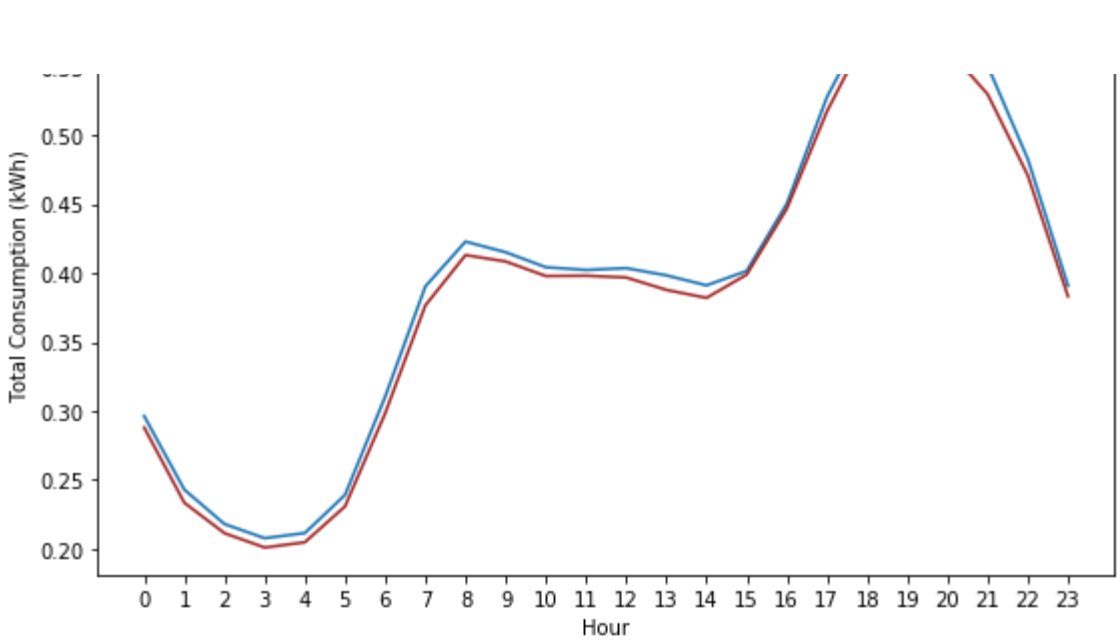

Monday Hourly Average Load
non-TOU
TOU
Total Consumption (kWh)
0.2
0.3
0.4
0.5
0.6
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

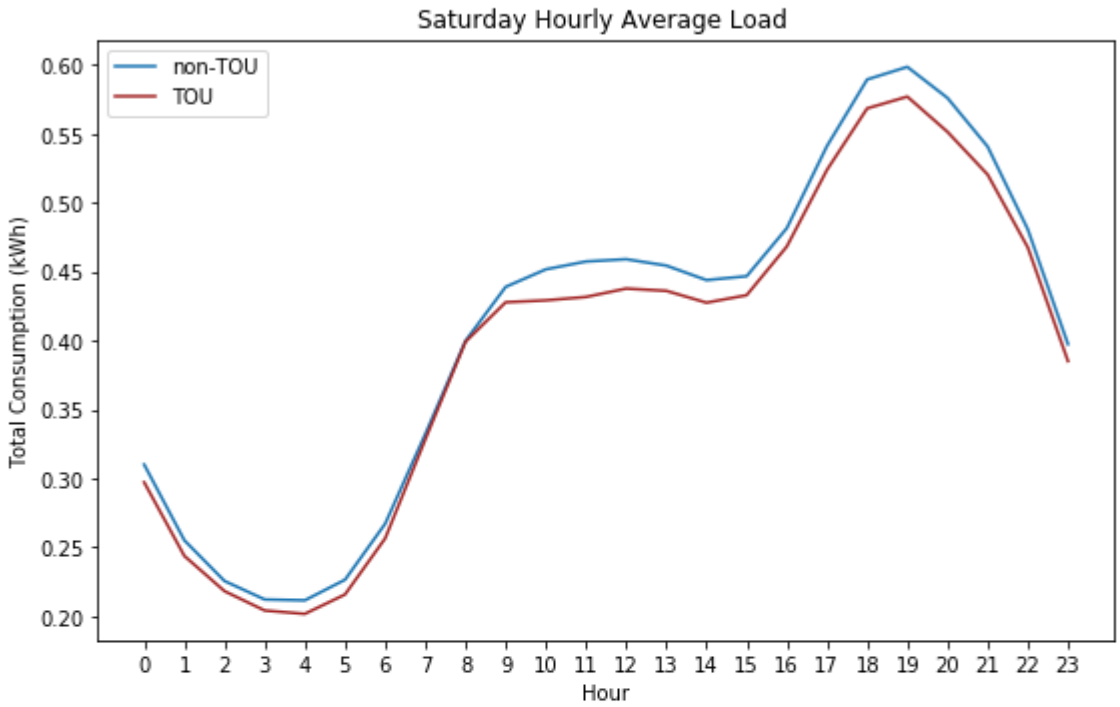

Tuesday Hourly Average Load
non-TOU
TOU
Total Consumption (kWh)
0.2
0.3
0.4
0.5
0.6
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

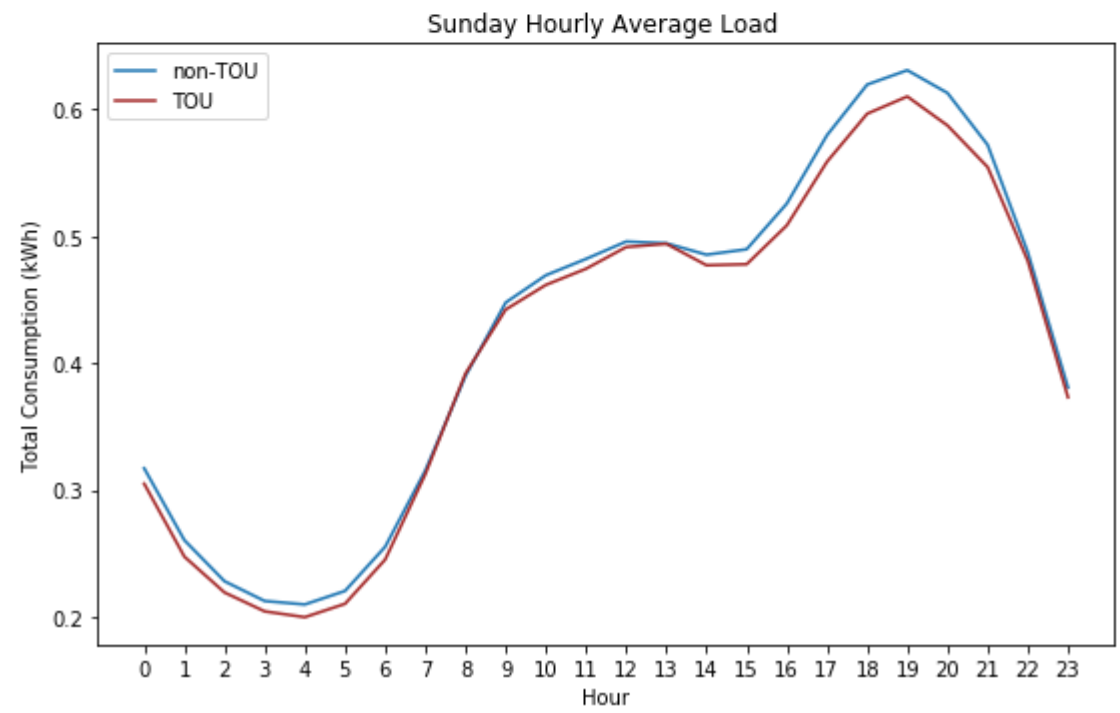

Wednesday Hourly Average Load
non-TOU
TOU
Total Consumption (kWh)
0.2
0.3
0.4
0.5
0.6
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour


Thursday Hourly Average Load
non-TOU
TOU
Total Consumption (kWh)
0.20
0.25
0.30
0.35
0.40
0.45
0.50
0.55
0.60
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour


Friday Hourly Average Load
non-TOU
TOU
0.55
0.60







Compare the two groups in the non-flexible time periods

It's worth mentioning that TOU tariffs:

high 0.672, default 0.1176, low 0.0399; non-TOU tariff is fixed 0.14228.

Read the tariff data and use it to filter out the time that has no event tags and compare the TOU data at those time periods with the corresponding non-TOU data. There are two things that need to pay attention.

1) The tariff data is starting from 1/1/2013 0:30, which means from 0:00 to 0:30 the price is listed in the row of 1/1/2013 0:30. This is kind of different from TOU data, which is listed at the begining of the 30min intervals. So we need to first convert it to be consistent with TOU timing format.

2) Since TOU is notified one day ahead, it would be better we could filter out the whole day data that have TOU events. The high, low rate may affect the demand schedule of that day instead of only when high, low rate appear.

Next, we need to filter out the data that belongs to any event data

In [22]:
```
test=datetime.strptime(df_tariff_1h.GMT[0], "%Y-%m-%d %H:%M:%S")
test.second
```

Out[22]: 0

In [67]:
```
# first, create a list of days that belongs to event days
event_days = set()
event_series = df_tariff_1h[df_tariff_1h.Event_tags.notnull()].GMT
for i in event_series:
    event_days.add(datetime.strptime(i[:10], "%Y-%m-%d" ).date()) # add all event dates to the set
df_help = pd.DataFrame(pd.to_datetime(df_tariff_1h.GMT).dt.date) #str to datatime and extract date then make
 it to dataframe
# df_help[df_help['GMT'].isin(event_days)] #this shows the event days
# we can use ~df_help['GMT'].isin(event_days) to generate any non-flexible period items

# create TOU and non-TOU demand data in non-flexible hours
df_wea1h_nf = df_wea1h[~df_help['GMT'].isin(event_days)]
df_Ntou1h_nf = df_Ntou1h[~df_help['GMT'].isin(event_days)]
df_tou1h_nf = df_tou1h[~df_help['GMT'].isin(event_days)]
```

In [9]:
```
df_tou1h_nf.head()
```

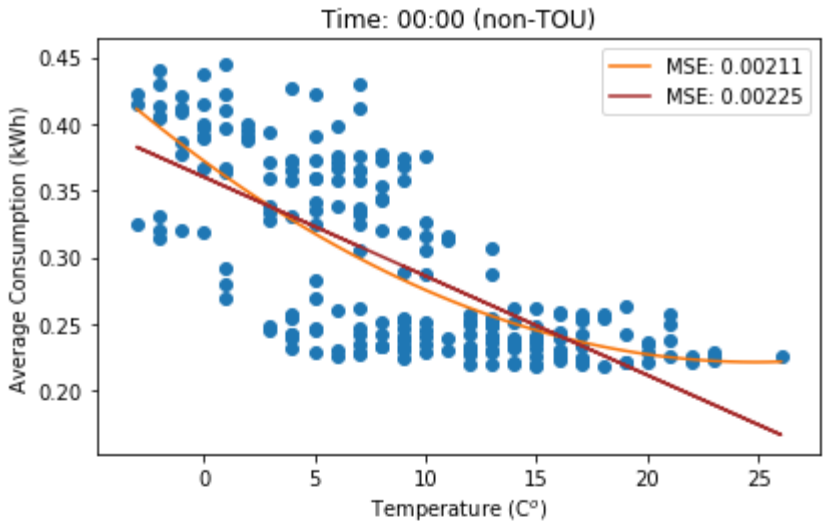
Out[9]:

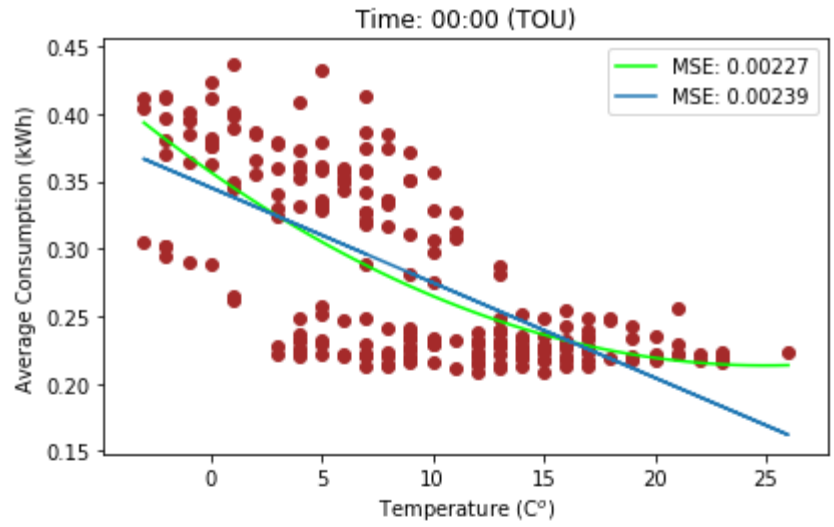
	GMT	D0000	D0001	D0002	D0003	D0004	D0005	D0006	D0007	D0008	...	D1015	D1016	D1017	D1018	D1019	D1020	D1021
0	2013-01-01 00:00:00	1.447	0.429	0.451	0.155	0.397	0.106	0.307	0.268	0.390	...	0.048	0.033	0.502	0.117	0.378	1.143	0.155
1	2013-01-01 01:00:00	0.336	0.377	0.467	0.153	0.227	0.109	0.365	0.300	0.528	...	0.062	0.010	0.528	0.117	0.399	0.856	0.098
2	2013-01-01 02:00:00	0.244	0.151	0.490	0.154	0.126	0.108	0.060	0.197	0.150	...	0.032	0.010	0.217	0.310	0.362	0.521	0.083
3	2013-01-01 03:00:00	0.213	0.155	0.580	0.153	0.057	0.109	0.067	0.167	0.151	...	0.062	0.011	0.206	0.119	0.368	0.470	0.113
4	2013-01-01 04:00:00	0.210	0.121	0.579	0.156	0.061	0.107	0.056	0.144	0.133	...	0.061	0.096	0.482	0.118	0.354	0.246	0.097

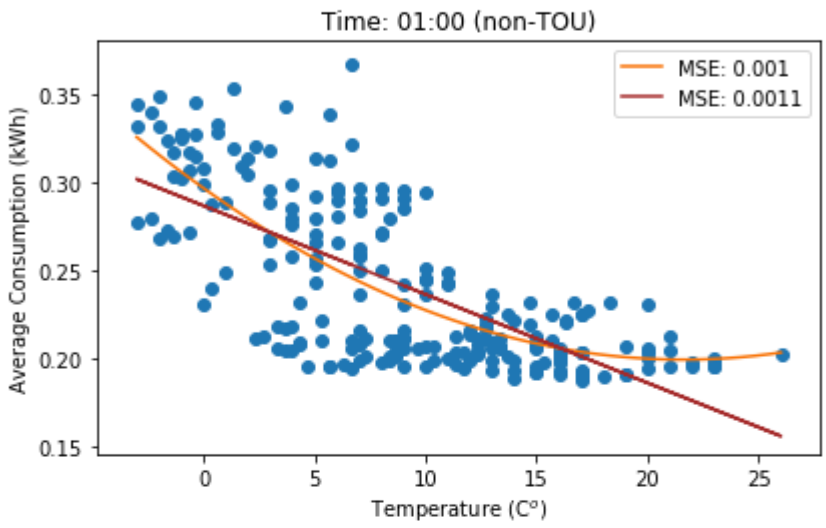
5 rows × 1026 columns

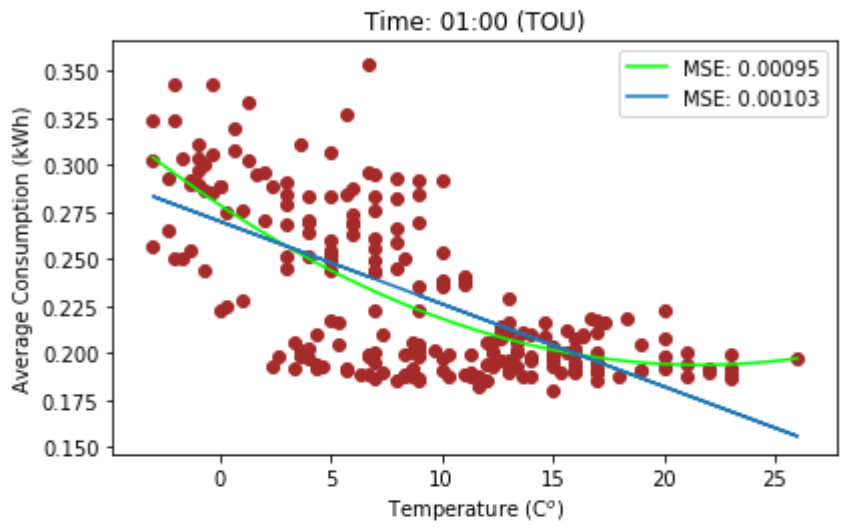
Now Let's plot the non-TOU vs TOU in terms of showing the elements in Average column such that they are hourly energy consumption v.s. temperature over non-event days

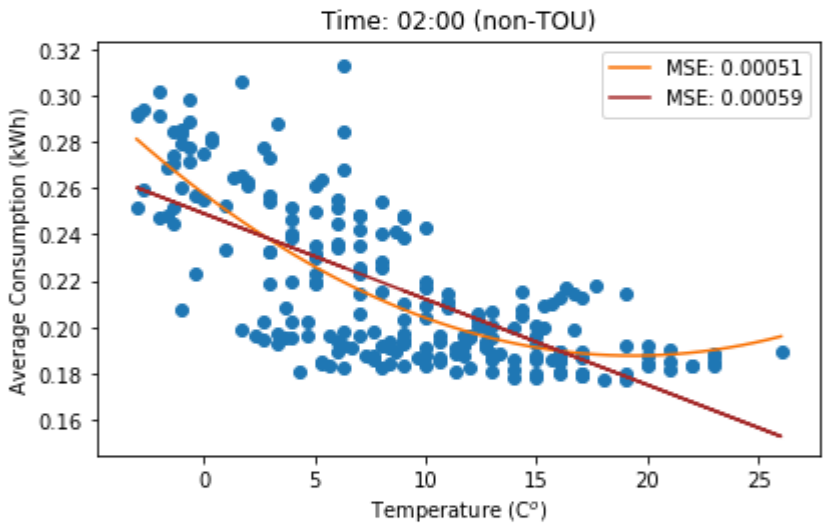
In [25]:
```
# Regression for non-event data
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
# non-TOU vs TOU in terms of showing the elements in Average column such that they are hourly energy consumpt
ion v.s. temperature over non-event days
fig_all = plt.figure(figsize = (12,90))
ax_Ntou = [] # store subplot objects
ax_tou = [] # store subplot objects
lm = LinearRegression()
for i in range(24):
    ax_Ntou.append(fig_all.add_subplot(24, 2, 2 * i + 1))
    if i <= 9:
        x = df_wea1h_nf.TempC[df_wea1h_nf.GMT.str.contains('0' + str(i) + ':00:00')].values
        y = df_Ntou1h_nf.Average[df_Ntou1h_nf.GMT.str.contains('0' + str(i) + ':00:00')].values
        xp = np.linspace(min(x), max(x), 50) # polynomial regression
        z = np.polyfit(x, y, 2)
        p = np.poly1d(z)
        ax_Ntou[-1].plot(xp, p(xp), color = 'xkcd:orange', label = 'MSE: ' + str(round(mean_squared_error(y,
p(x)), 5)))
        x = x.reshape(-1, 1)
        y = y.reshape(-1, 1)
        lm.fit(x, y)
        ax_Ntou[-1].plot(x, lm.predict(x), color = 'brown', label = 'MSE: ' + str(round(mean_squared_error(y,
lm.predict(x)), 5)))
        ax_Ntou[-1].scatter(x, y)
        ax_Ntou[-1].set_title('Time: 0' + str(i) + ':00 (non-TOU)')
        ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_Ntou[-1].set_ylabel('Average Consumption (kWh)')
        ax_Ntou[-1].legend()
    else:
        x = df_wea1h_nf.TempC[df_wea1h_nf.GMT.str.contains(str(i) + ':00:00')].values
        y = df_Ntou1h_nf.Average[df_Ntou1h_nf.GMT.str.contains(str(i) + ':00:00')].values
        xp = np.linspace(min(x), max(x), 50) # polynomial regression
        z = np.polyfit(x, y, 2)
        p = np.poly1d(z)
        ax_Ntou[-1].plot(xp, p(xp), color = 'xkcd:orange', label = 'MSE: ' + str(round(mean_squared_error(y,
p(x)), 5)))
        x = x.reshape(-1, 1)
        y = y.reshape(-1, 1)
        lm.fit(x, y)
        ax_Ntou[-1].plot(x, lm.predict(x), color = 'brown', label = 'MSE: ' + str(round(mean_squared_error(y,
lm.predict(x)), 5)))
        ax_Ntou[-1].scatter(x, y)
        ax_Ntou[-1].set_title('Time: ' + str(i) + ':00 (non-TOU)')
        ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_Ntou[-1].set_ylabel('Average Consumption (kWh)')
        ax_Ntou[-1].legend()
for i in range(24):
    ax_tou.append(fig_all.add_subplot(24, 2, 2 * i + 2))
    if i <= 9:
        x = df_wea1h_nf.TempC[df_wea1h_nf.GMT.str.contains('0' + str(i) + ':00:00')].values
        y = df_tou1h_nf.Average[df_tou1h_nf.GMT.str.contains('0' + str(i) + ':00:00')].values
        xp = np.linspace(min(x), max(x), 50) # polynomial regression
        z = np.polyfit(x, y, 2)
        p = np.poly1d(z)
        ax_tou[-1].plot(xp, p(xp), color = 'lime', label = 'MSE: ' + str(round(mean_squared_error(y, p(x)), 5
)))
        x = x.reshape(-1, 1)
        y = y.reshape(-1, 1)
        lm.fit(x, y)
        ax_tou[-1].plot(x, lm.predict(x), label = 'MSE: ' + str(round(mean_squared_error(y, lm.predict(x)), 5
)))
        ax_tou[-1].scatter(x, y, color = 'brown')
        ax_tou[-1].set_title('Time: 0' + str(i) + ':00 (TOU)')
        ax_tou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_tou[-1].set_ylabel('Average Consumption (kWh)')
        ax_tou[-1].legend()
    else:
        x = df_wea1h_nf.TempC[df_wea1h_nf.GMT.str.contains(str(i) + ':00:00')].values
        y = df_tou1h_nf.Average[df_tou1h_nf.GMT.str.contains(str(i) + ':00:00')].values
        xp = np.linspace(min(x), max(x), 50) # polynomial regression
        z = np.polyfit(x, y, 2)
        p = np.poly1d(z)
        ax_tou[-1].plot(xp, p(xp), color = 'lime', label = 'MSE: ' + str(round(mean_squared_error(y, p(x)), 5
)))
        x = x.reshape(-1, 1)
        y = y.reshape(-1, 1)
        lm.fit(x, y)
        ax_tou[-1].plot(x, lm.predict(x), label = 'MSE: ' + str(round(mean_squared_error(y, lm.predict(x)), 5
)))
        ax_tou[-1].scatter(x, y, color = 'brown')
        ax_tou[-1].set_title('Time: ' + str(i) + ':00 (TOU)')
        ax_tou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_tou[-1].set_ylabel('Average Consumption (kWh)')
        ax_tou[-1].legend()
plt.tight_layout()
```

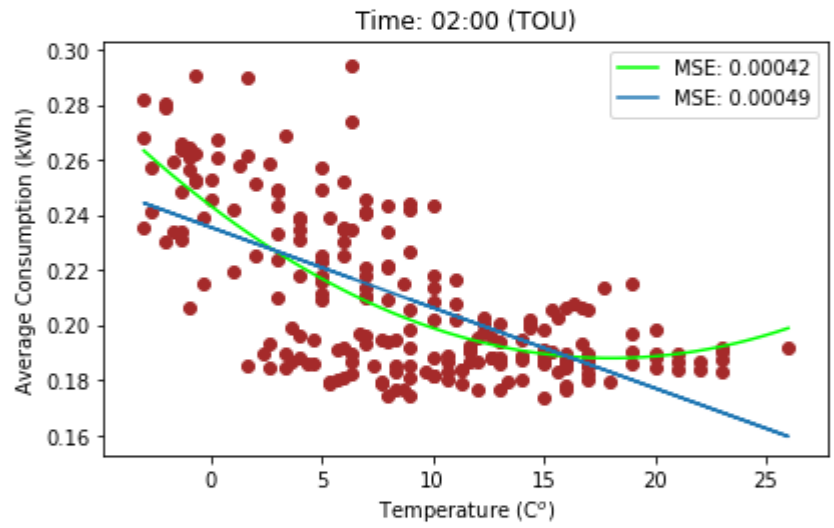

Time: 00:00 (non-TOU)
MSE: 0.00211
MSE: 0.00225
Average Consumption (kWh)
Temperature (C°)

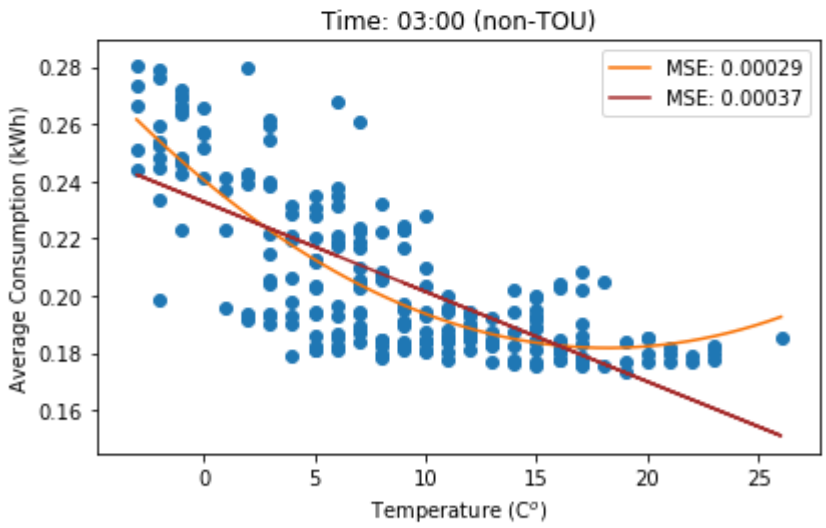

Time: 00:00 (TOU)
MSE: 0.00227
MSE: 0.00239
Average Consumption (kWh)
Temperature (C°)

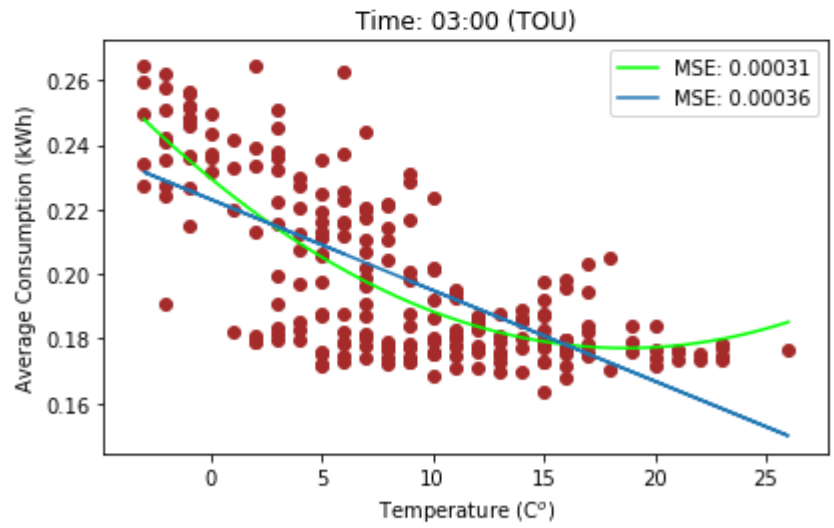

Time: 01:00 (non-TOU)
MSE: 0.001
MSE: 0.0011
Average Consumption (kWh)
Temperature (C°)

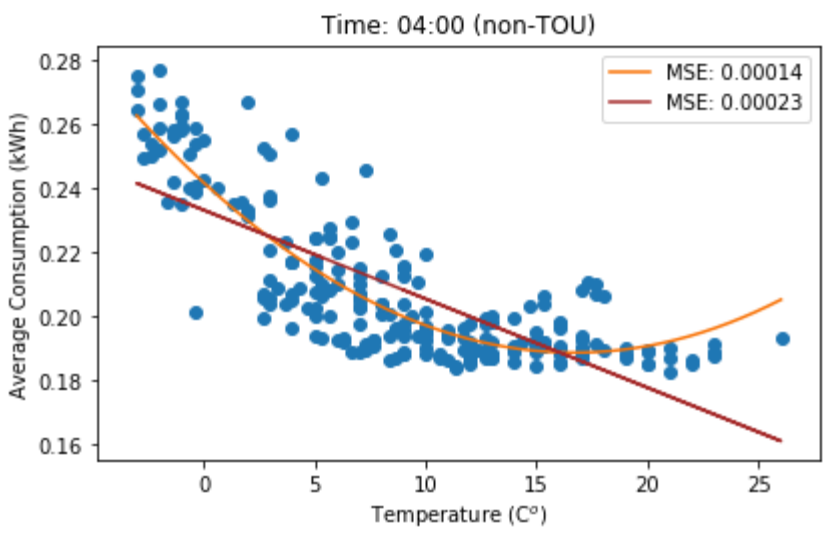

Time: 01:00 (TOU)
MSE: 0.00095
MSE: 0.00103
Average Consumption (kWh)
Temperature (C°)

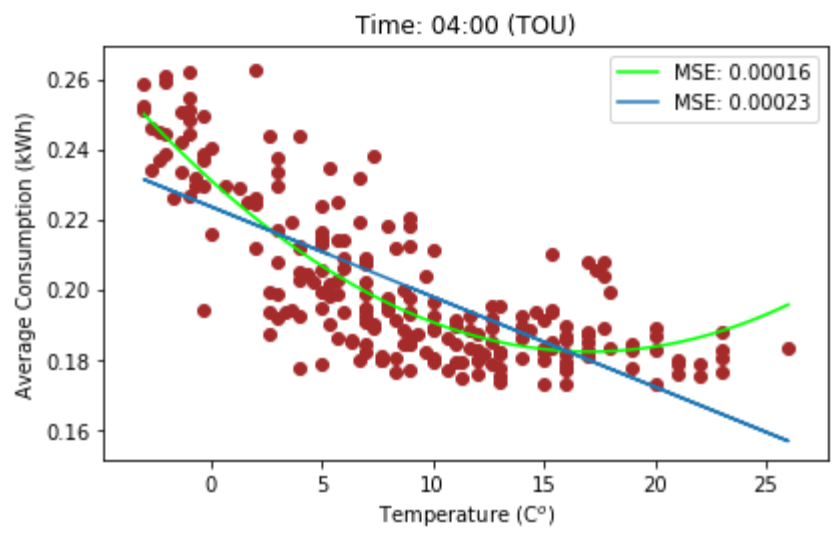

Time: 02:00 (non-TOU)
MSE: 0.00051
MSE: 0.00059
Average Consumption (kWh)
Temperature (C°)


Time: 02:00 (TOU)
MSE: 0.00042
MSE: 0.00049
Average Consumption (kWh)
Temperature (C°)


Time: 03:00 (non-TOU)
MSE: 0.00029
MSE: 0.00037
Average Consumption (kWh)
Temperature (C°)

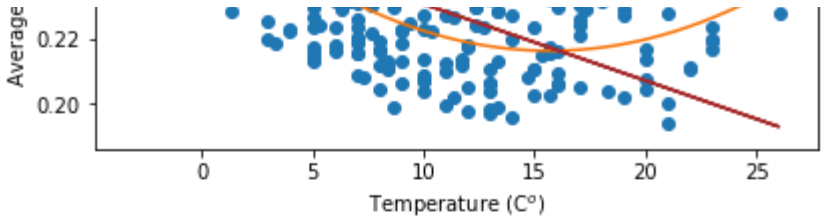

Time: 03:00 (TOU)
MSE: 0.00031
MSE: 0.00036
Average Consumption (kWh)
Temperature (C°)

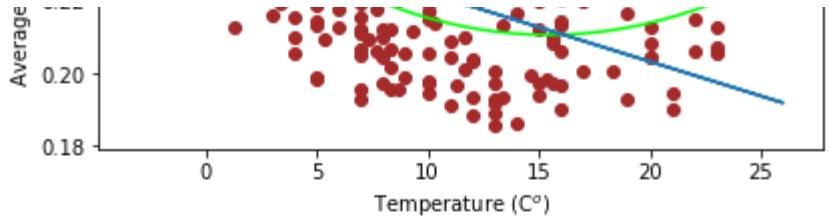

Time: 04:00 (non-TOU)
MSE: 0.00014
MSE: 0.00023
Average Consumption (kWh)
Temperature (C°)

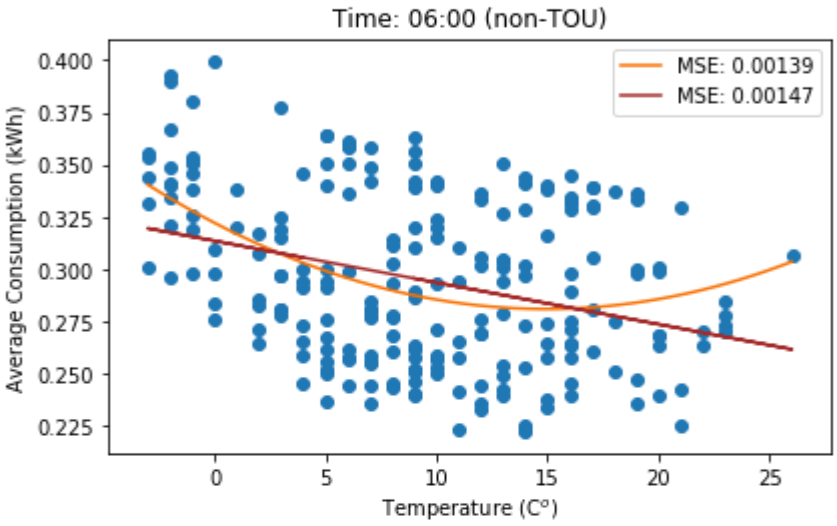

Time: 04:00 (TOU)
MSE: 0.00016
MSE: 0.00023
Average Consumption (kWh)
Temperature (C°)

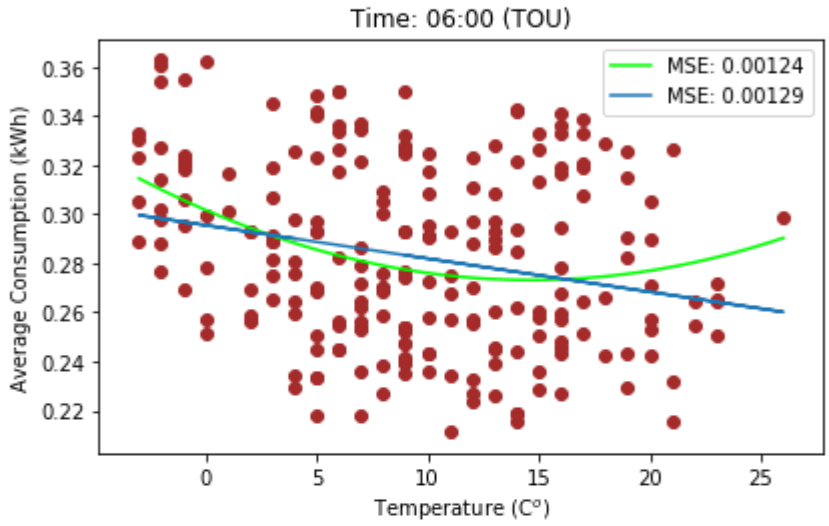
Time: 05:00 (non-TOU)
MSE: 0.00023
MSE: 0.00032
Temperature (C°)

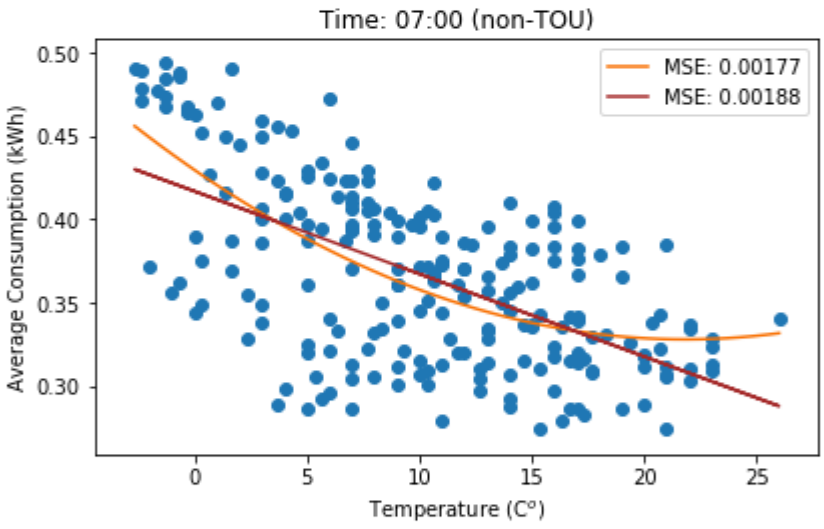
Time: 05:00 (TOU)
MSE: 0.00023
MSE: 0.00029
Temperature (C°)

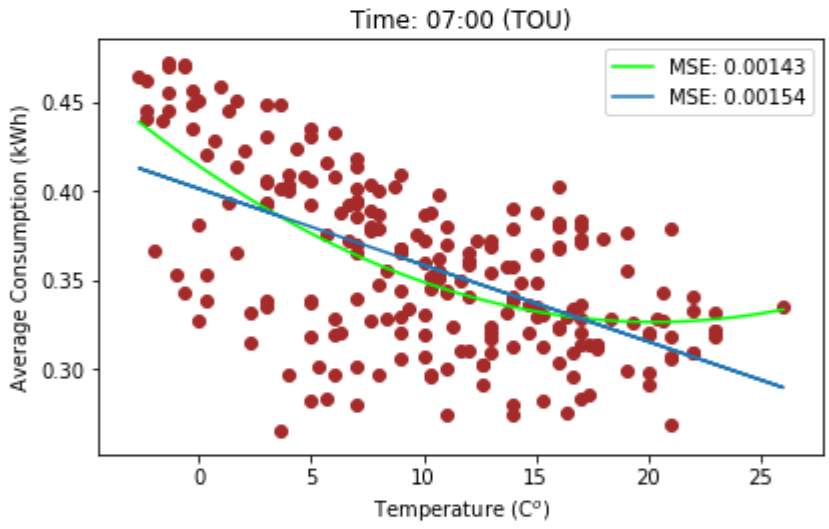

Average
0.22
0.20
0
5
10
15
20
25
Temperature (C°)

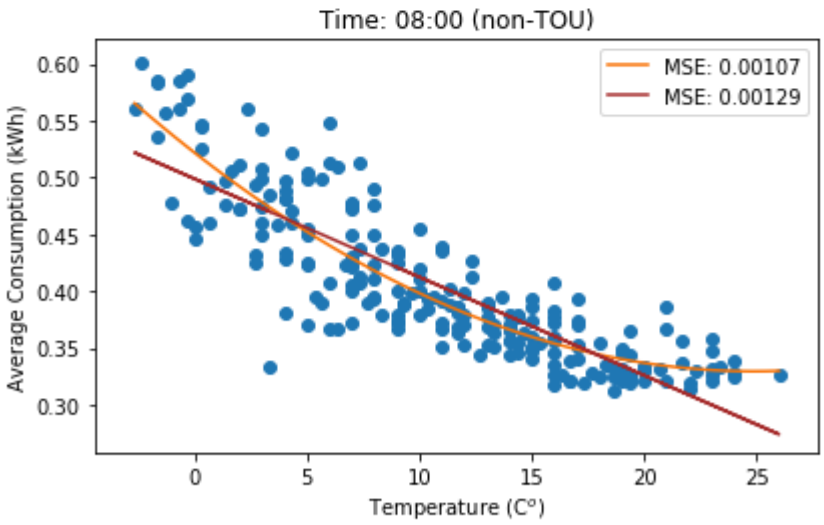

Average
0.20
0.18
0
5
10
15
20
25
Temperature (C°)

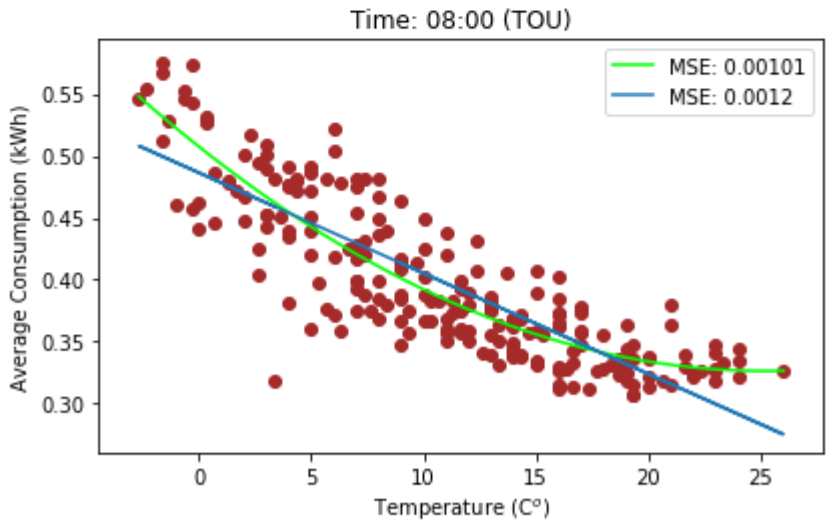

Time: 06:00 (non-TOU)
MSE: 0.00139
MSE: 0.00147
Average Consumption (kWh)
0.400
0.375
0.350
0.325
0.300
0.275
0.250
0.225
0
5
10
15
20
25
Temperature (C°)

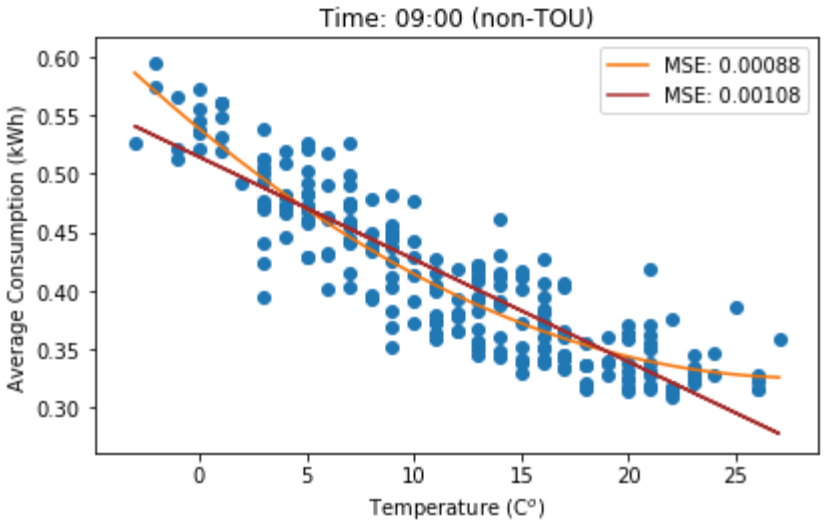

Time: 06:00 (TOU)
MSE: 0.00124
MSE: 0.00129
Average Consumption (kWh)
0.36
0.34
0.32
0.30
0.28
0.26
0.24
0.22
0
5
10
15
20
25
Temperature (C°)

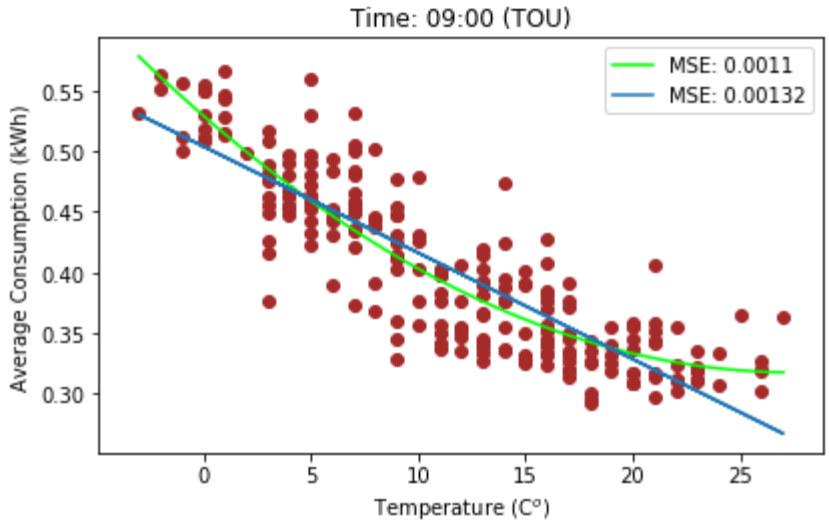

Time: 07:00 (non-TOU)
MSE: 0.00177
MSE: 0.00188
Average Consumption (kWh)
0.50
0.45
0.40
0.35
0.30
0
5
10
15
20
25
Temperature (C°)

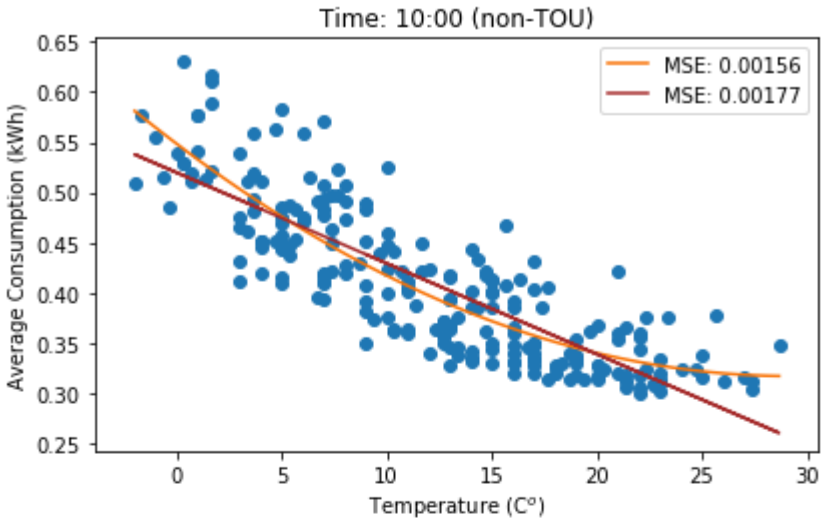

Time: 07:00 (TOU)
MSE: 0.00143
MSE: 0.00154
Average Consumption (kWh)
0.45
0.40
0.35
0.30
0
5
10
15
20
25
Temperature (C°)

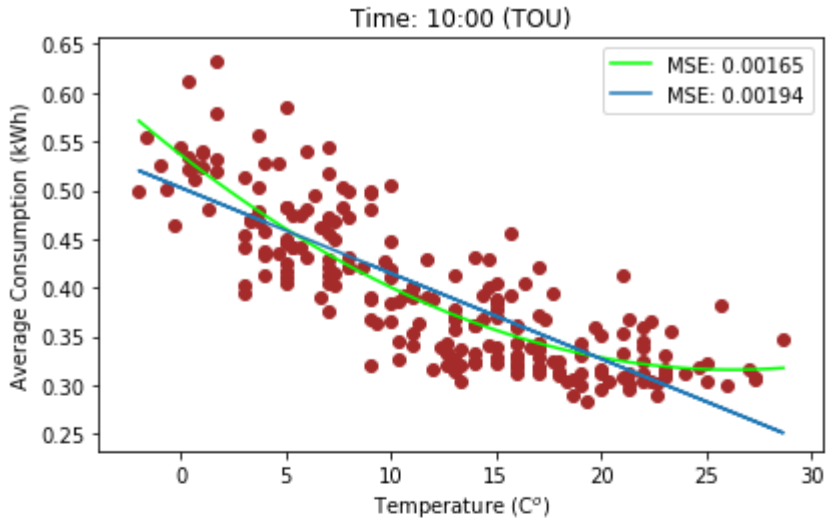

Time: 08:00 (non-TOU)
MSE: 0.00107
MSE: 0.00129
Average Consumption (kWh)
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0
5
10
15
20
25
Temperature (C°)


Time: 08:00 (TOU)
MSE: 0.00101
MSE: 0.0012
Average Consumption (kWh)
0.55
0.50
0.45
0.40
0.35
0.30
0
5
10
15
20
25
Temperature (C°)


Time: 09:00 (non-TOU)
MSE: 0.00088
MSE: 0.00108
Average Consumption (kWh)
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0
5
10
15
20
25
Temperature (C°)

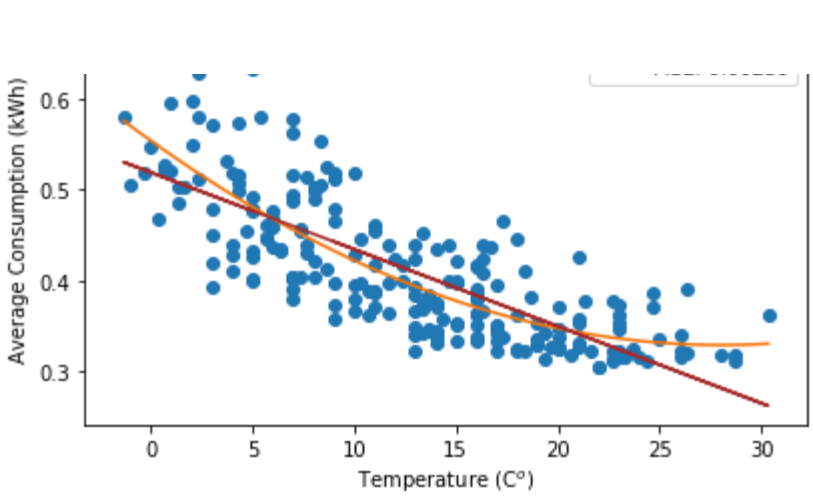

Time: 09:00 (TOU)
MSE: 0.0011
MSE: 0.00132
Average Consumption (kWh)
0.55
0.50
0.45
0.40
0.35
0.30
0
5
10
15
20
25
Temperature (C°)

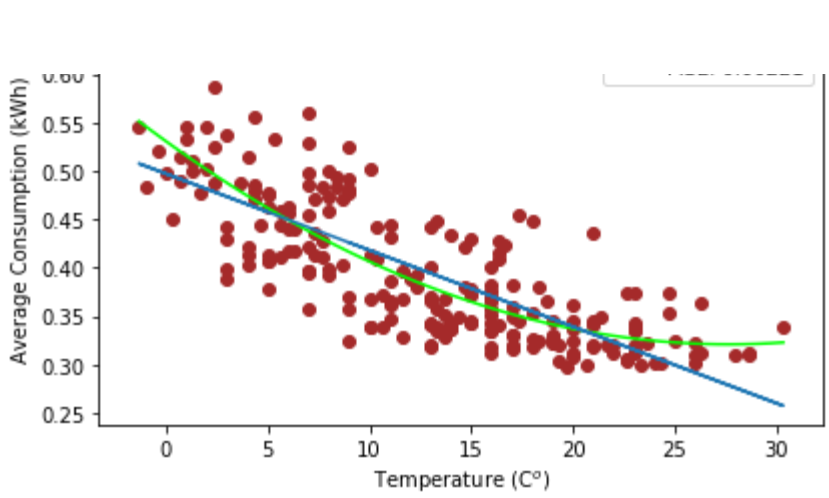

Time: 10:00 (non-TOU)
MSE: 0.00156
MSE: 0.00177
Average Consumption (kWh)
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0
5
10
15
20
25
30
Temperature (C°)

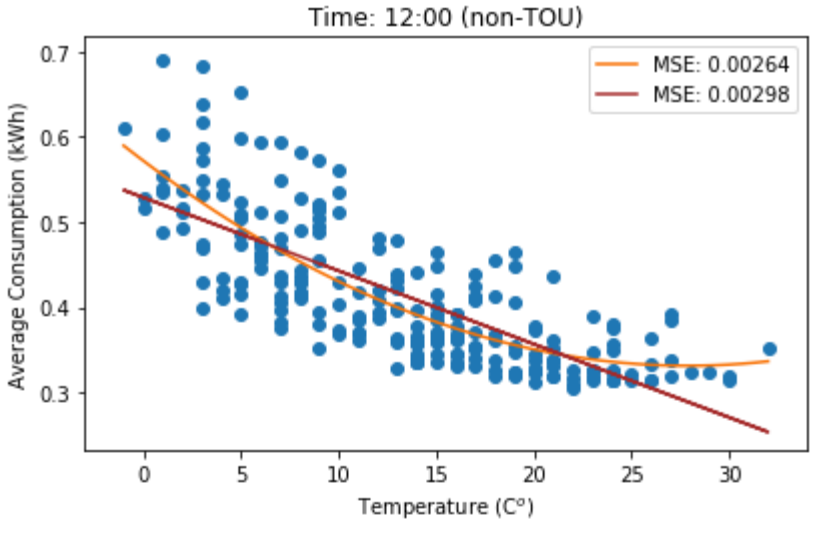

Time: 10:00 (TOU)
MSE: 0.00165
MSE: 0.00194
Average Consumption (kWh)
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0
5
10
15
20
25
30
Temperature (C°)

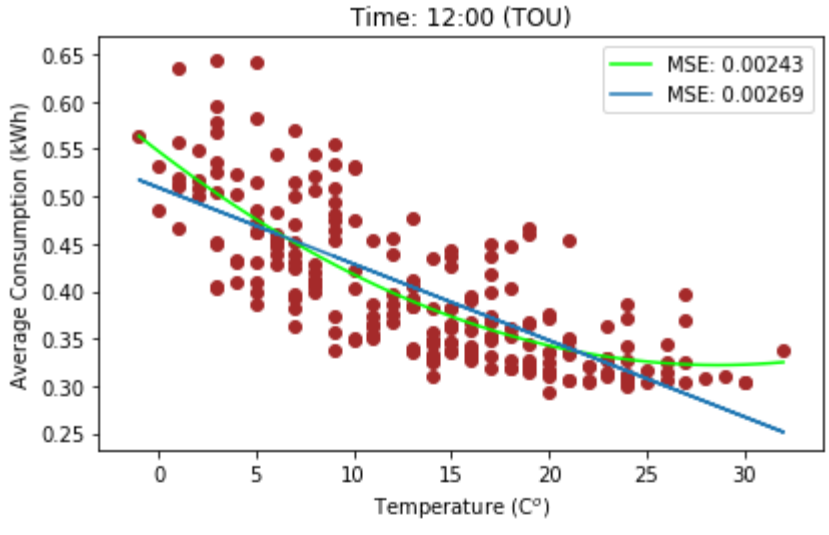
Time: 11:00 (non-TOU)
MSE: 0.00213
MSE: 0.00239

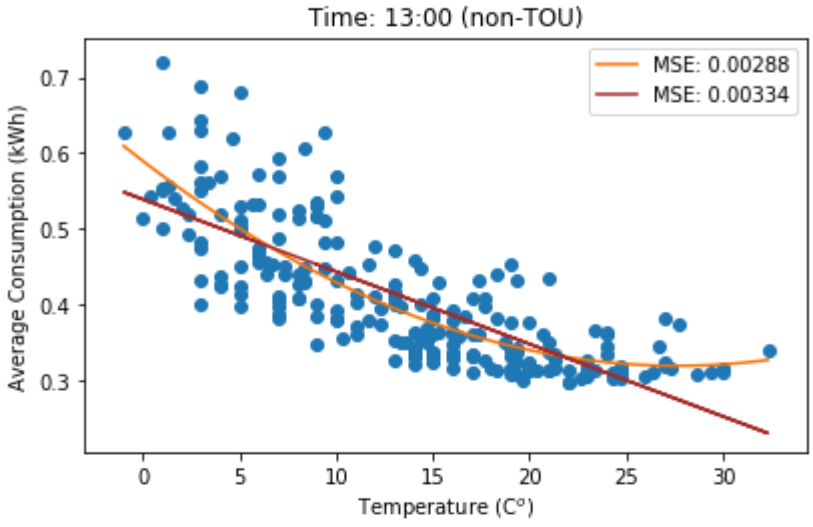
Time: 11:00 (TOU)
MSE: 0.00197
MSE: 0.00221

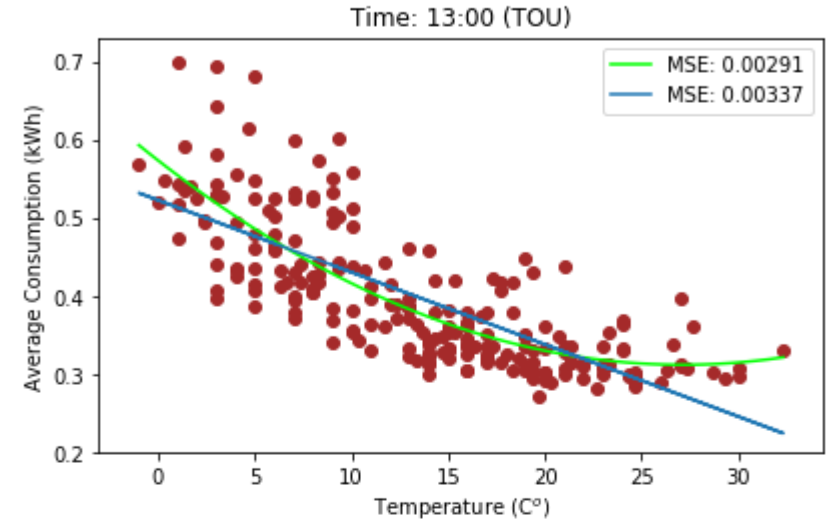

Average Consumption (kWh)
Temperature (C°)

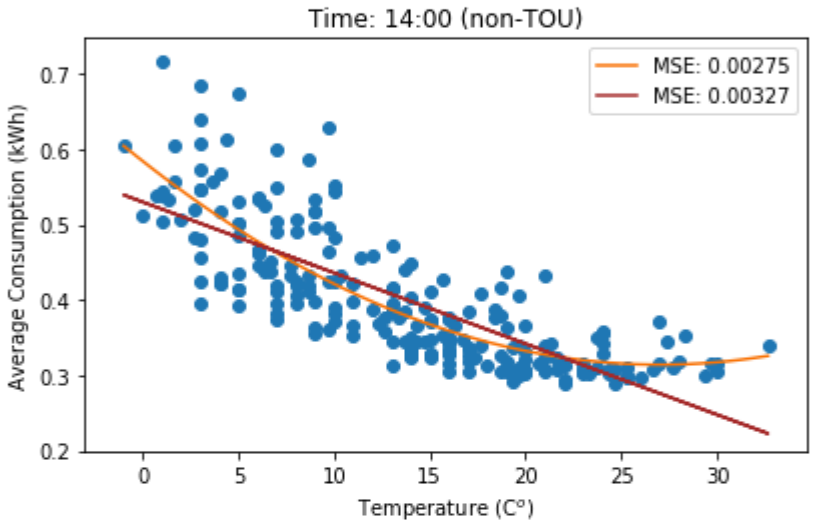

Average Consumption (kWh)
Temperature (C°)

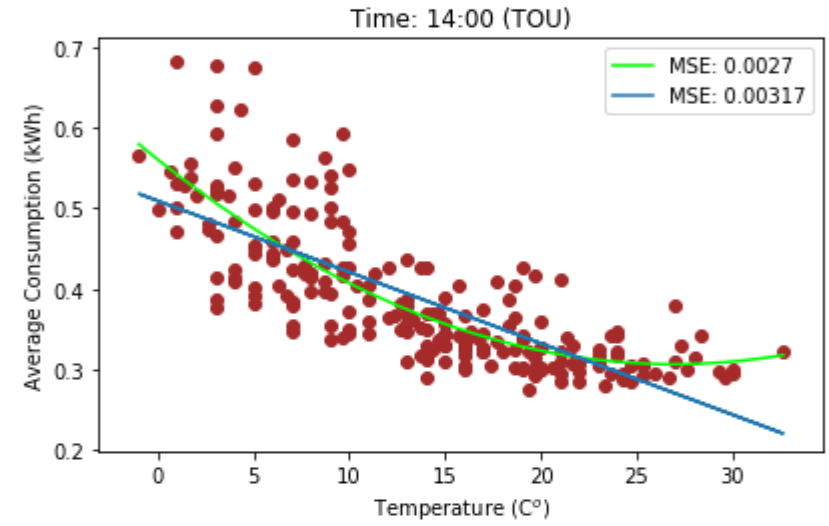

Time: 12:00 (non-TOU)
MSE: 0.00264
MSE: 0.00298
Average Consumption (kWh)
Temperature (C°)

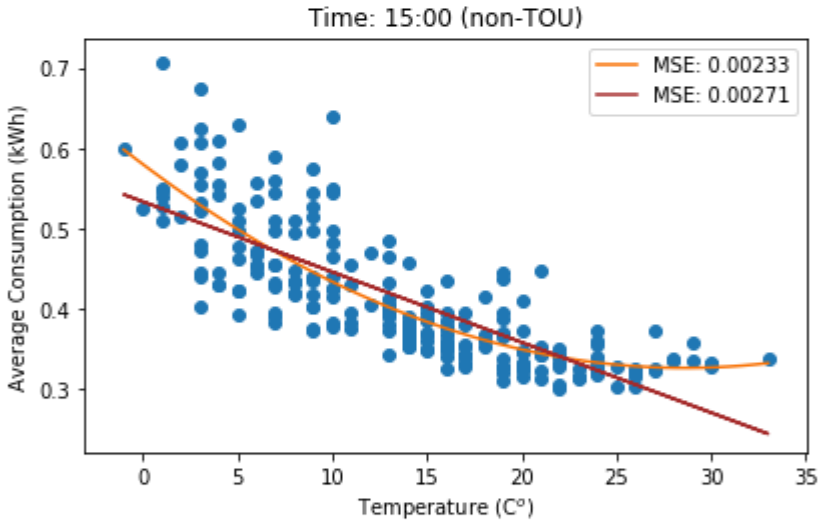

Time: 12:00 (TOU)
MSE: 0.00243
MSE: 0.00269
Average Consumption (kWh)
Temperature (C°)

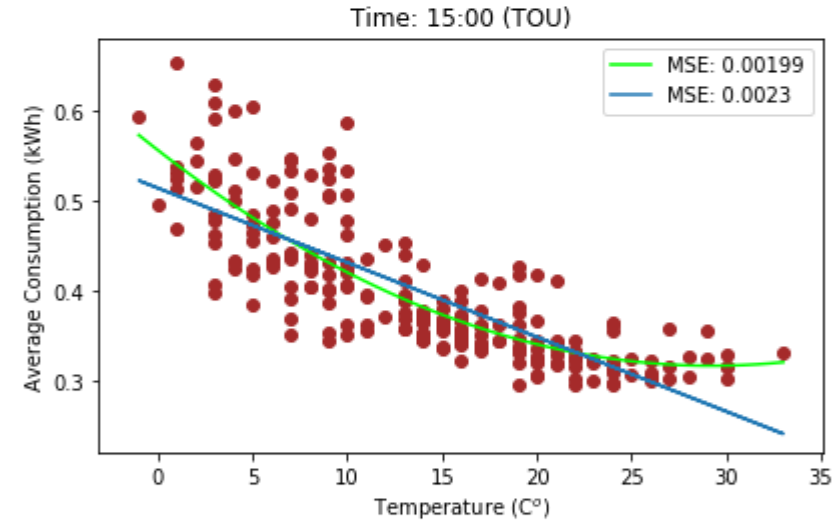

Time: 13:00 (non-TOU)
MSE: 0.00288
MSE: 0.00334
Average Consumption (kWh)
Temperature (C°)

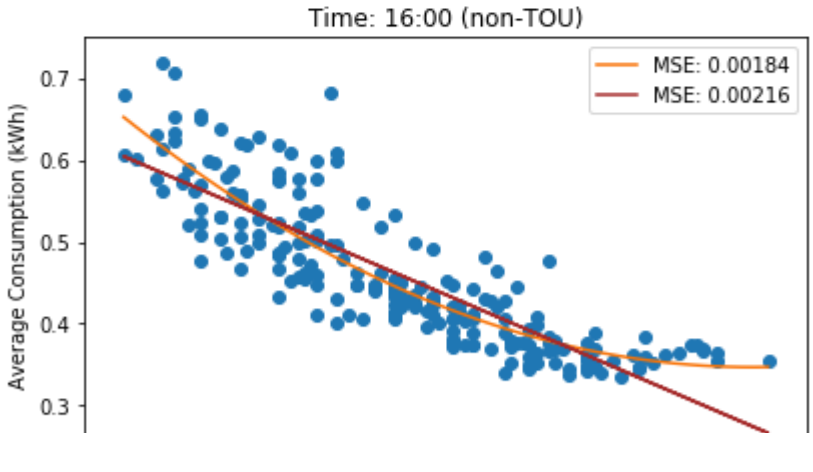

Time: 13:00 (TOU)
MSE: 0.00291
MSE: 0.00337
Average Consumption (kWh)
Temperature (C°)

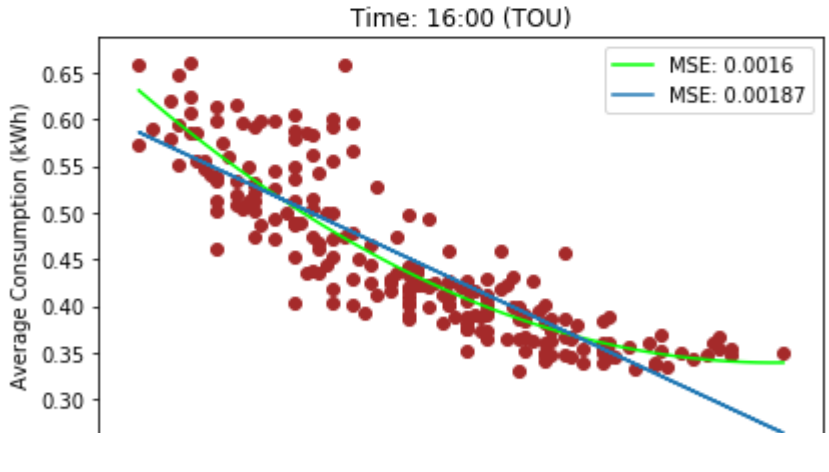

Time: 14:00 (non-TOU)
MSE: 0.00275
MSE: 0.00327
Average Consumption (kWh)
Temperature (C°)

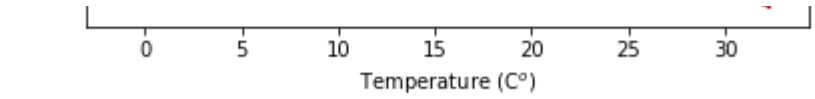

Time: 14:00 (TOU)
MSE: 0.0027
MSE: 0.00317
Average Consumption (kWh)
Temperature (C°)

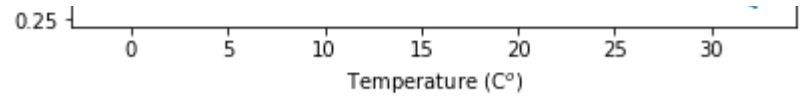

Time: 15:00 (non-TOU)
MSE: 0.00233
MSE: 0.00271
Average Consumption (kWh)
Temperature (C°)

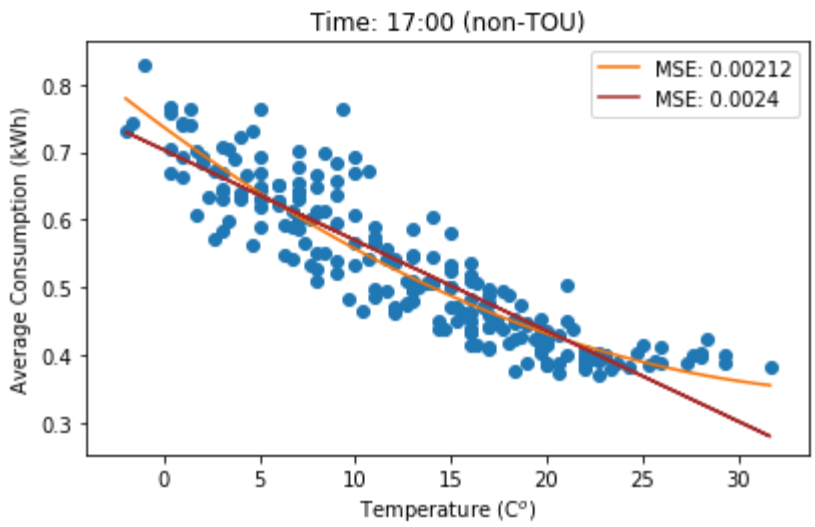

Time: 15:00 (TOU)
MSE: 0.00199
MSE: 0.0023
Average Consumption (kWh)
Temperature (C°)

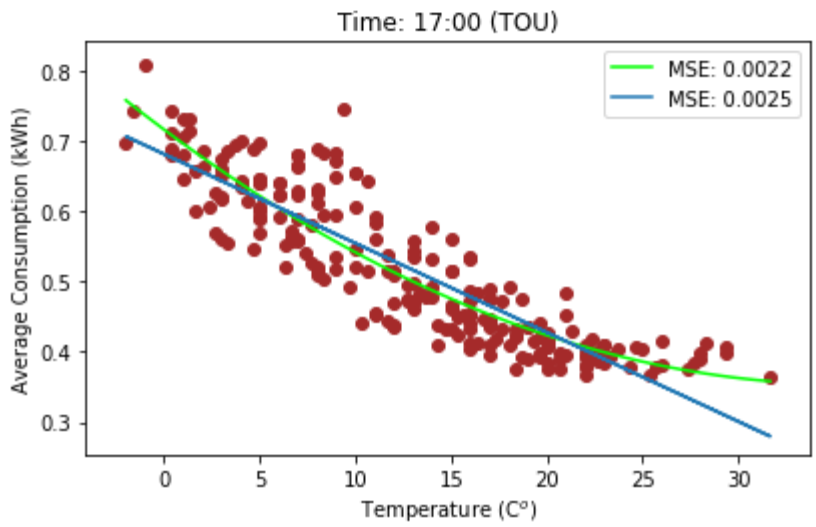

Time: 16:00 (non-TOU)
MSE: 0.00184
MSE: 0.00216
Average Consumption (kWh)

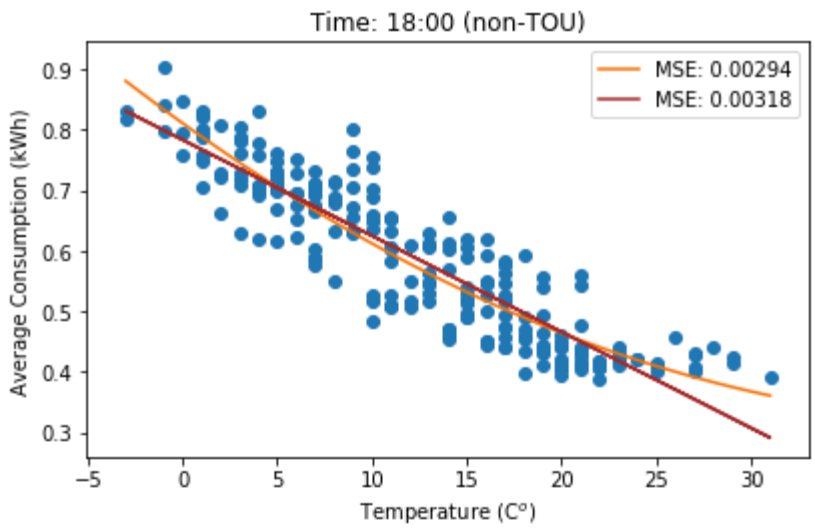

Time: 16:00 (TOU)
MSE: 0.0016
MSE: 0.00187
Average Consumption (kWh)

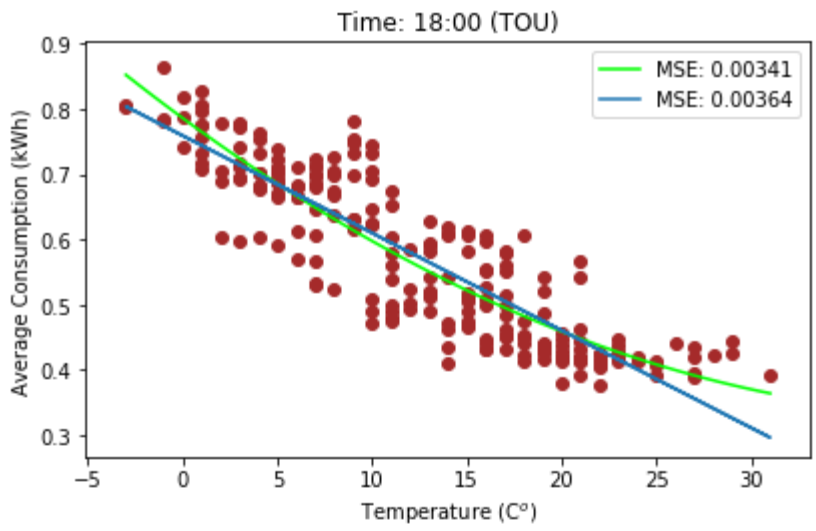

Temperature (C°)

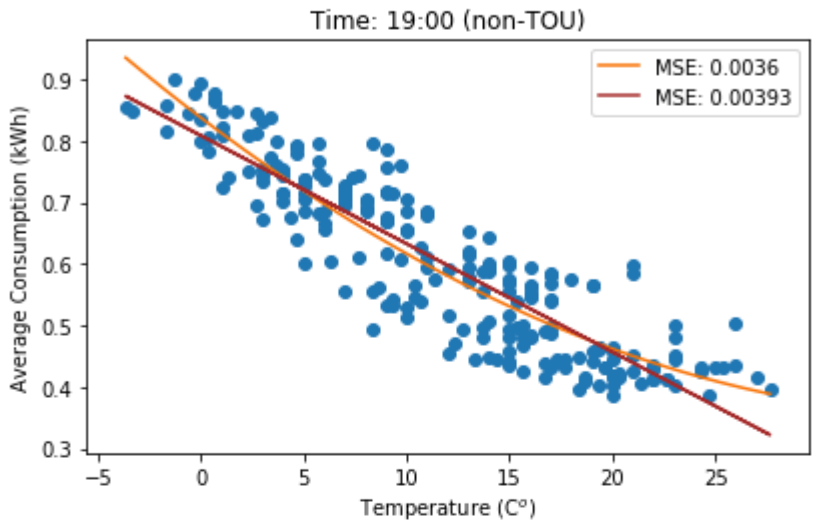

0.25
Temperature (C°)

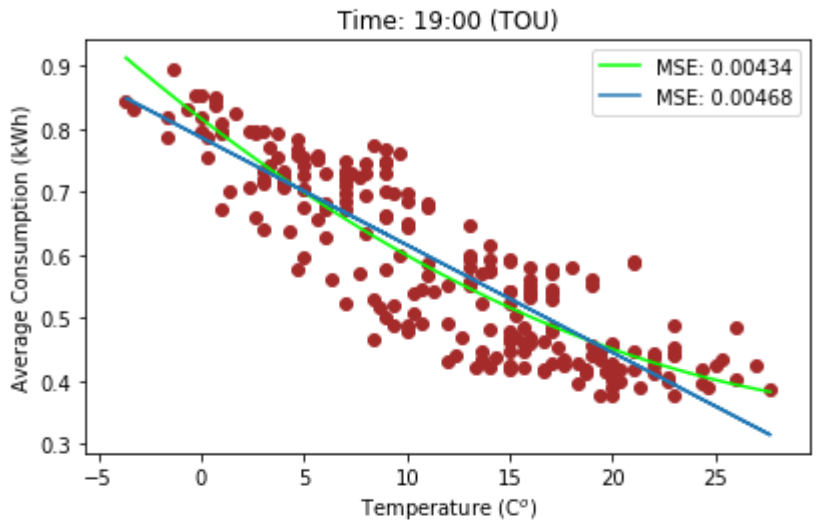

Time: 17:00 (non-TOU)
MSE: 0.00212
MSE: 0.0024
Average Consumption (kWh)
Temperature (C°)

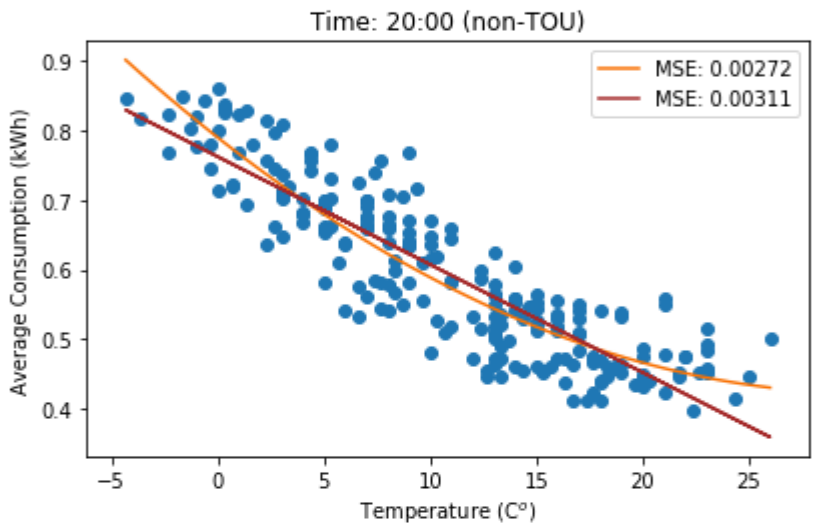

Time: 17:00 (TOU)
MSE: 0.0022
MSE: 0.0025
Average Consumption (kWh)
Temperature (C°)

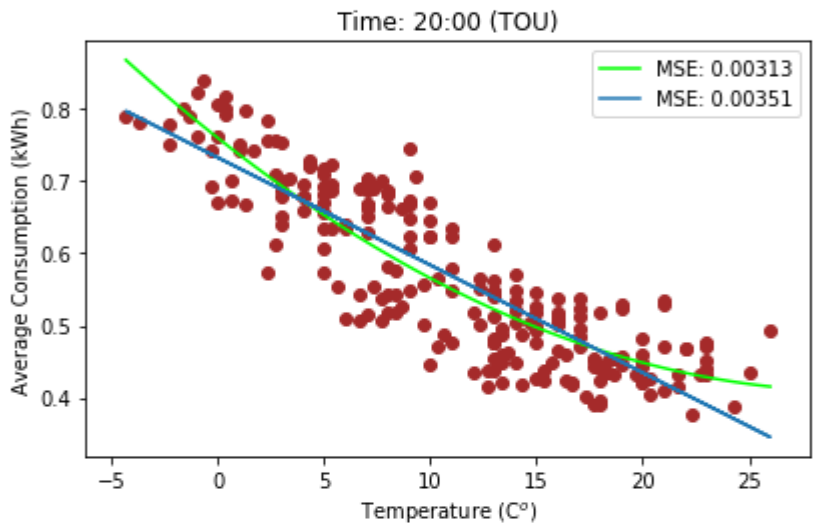

Time: 18:00 (non-TOU)
MSE: 0.00294
MSE: 0.00318
Average Consumption (kWh)
Temperature (C°)

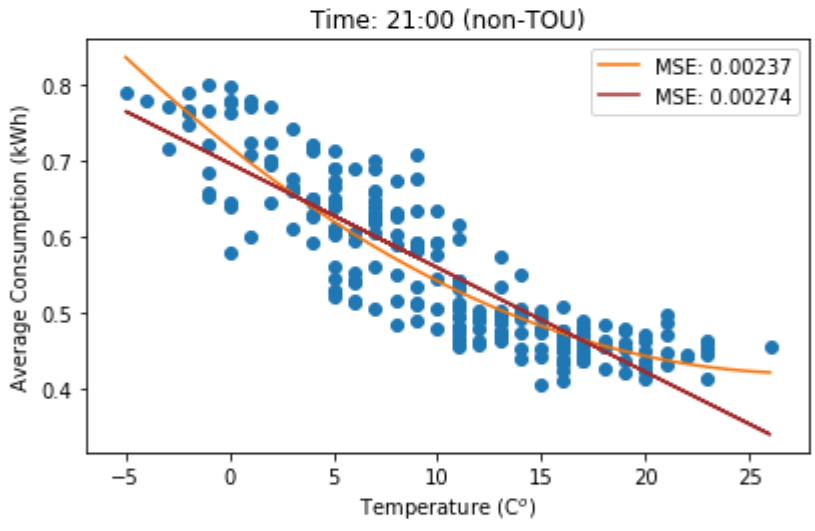

Time: 18:00 (TOU)
MSE: 0.00341
MSE: 0.00364
Average Consumption (kWh)
Temperature (C°)

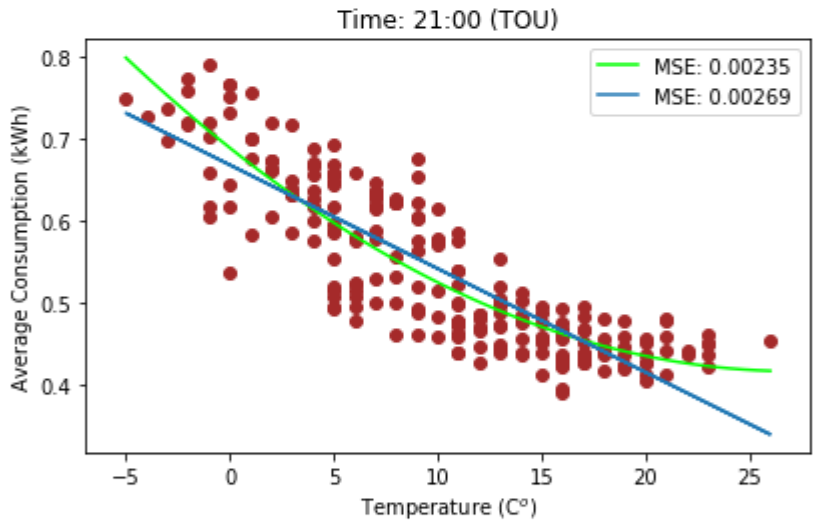

Time: 19:00 (non-TOU)
MSE: 0.0036
MSE: 0.00393
Average Consumption (kWh)
Temperature (C°)

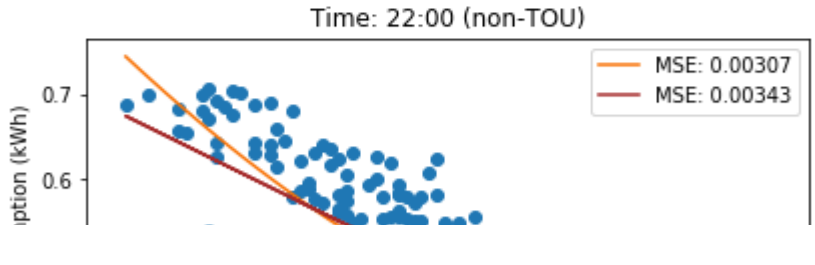

Time: 19:00 (TOU)
MSE: 0.00434
MSE: 0.00468
Average Consumption (kWh)
Temperature (C°)

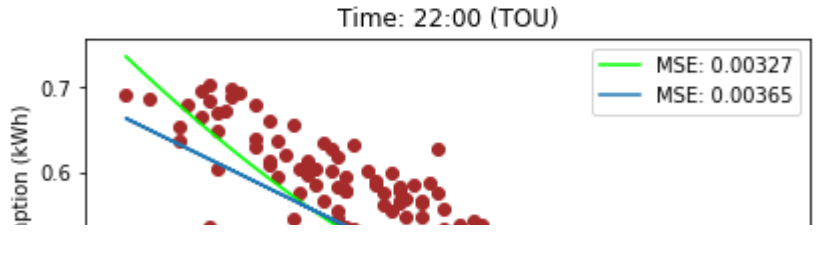

Time: 20:00 (non-TOU)
MSE: 0.00272
MSE: 0.00311
Average Consumption (kWh)
Temperature (C°)

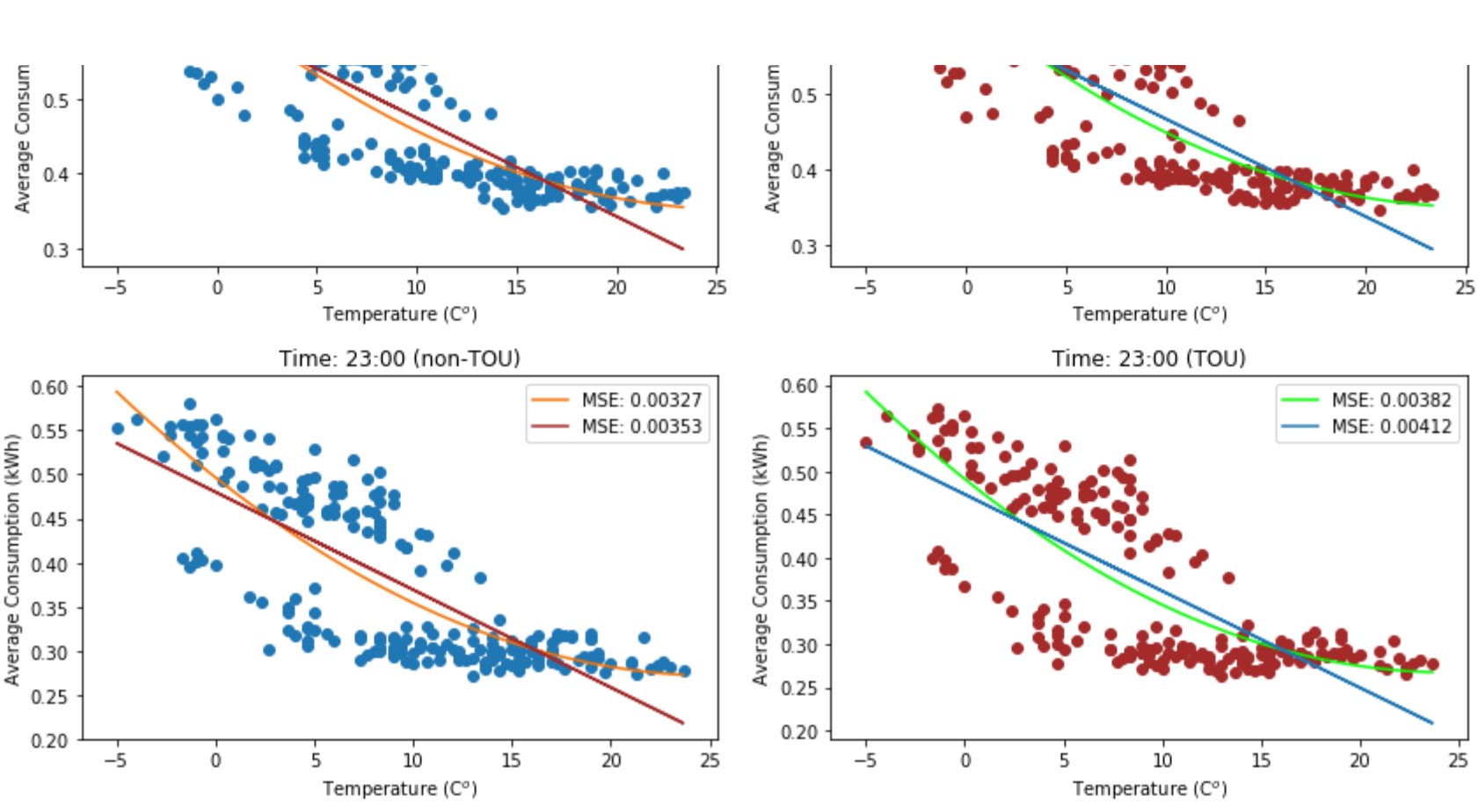

Time: 20:00 (TOU)
MSE: 0.00313
MSE: 0.00351
Average Consumption (kWh)
Temperature (C°)


Time: 21:00 (non-TOU)
MSE: 0.00237
MSE: 0.00274
Average Consumption (kWh)
Temperature (C°)

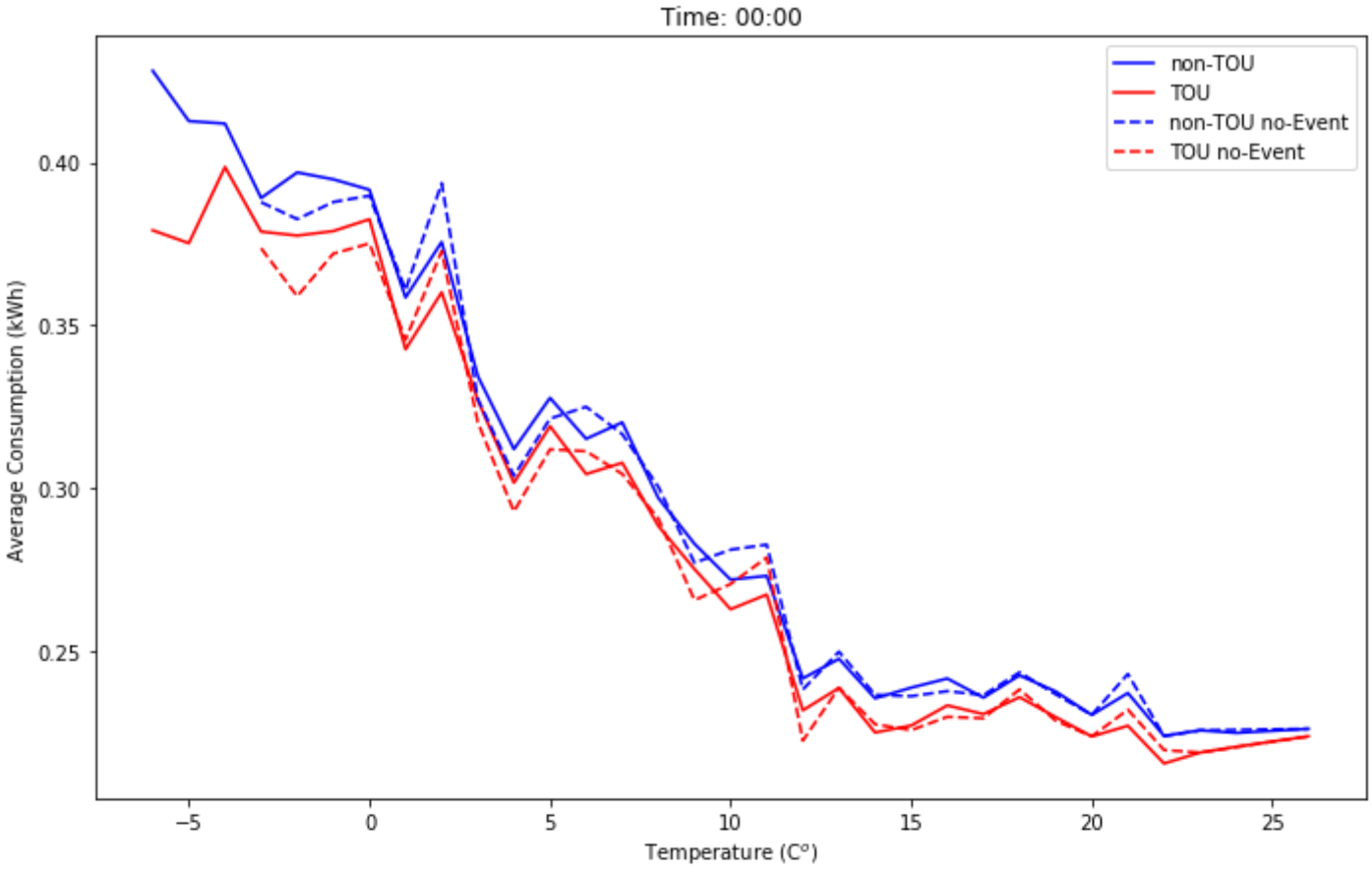

Time: 21:00 (TOU)
MSE: 0.00235
MSE: 0.00269
Average Consumption (kWh)
Temperature (C°)

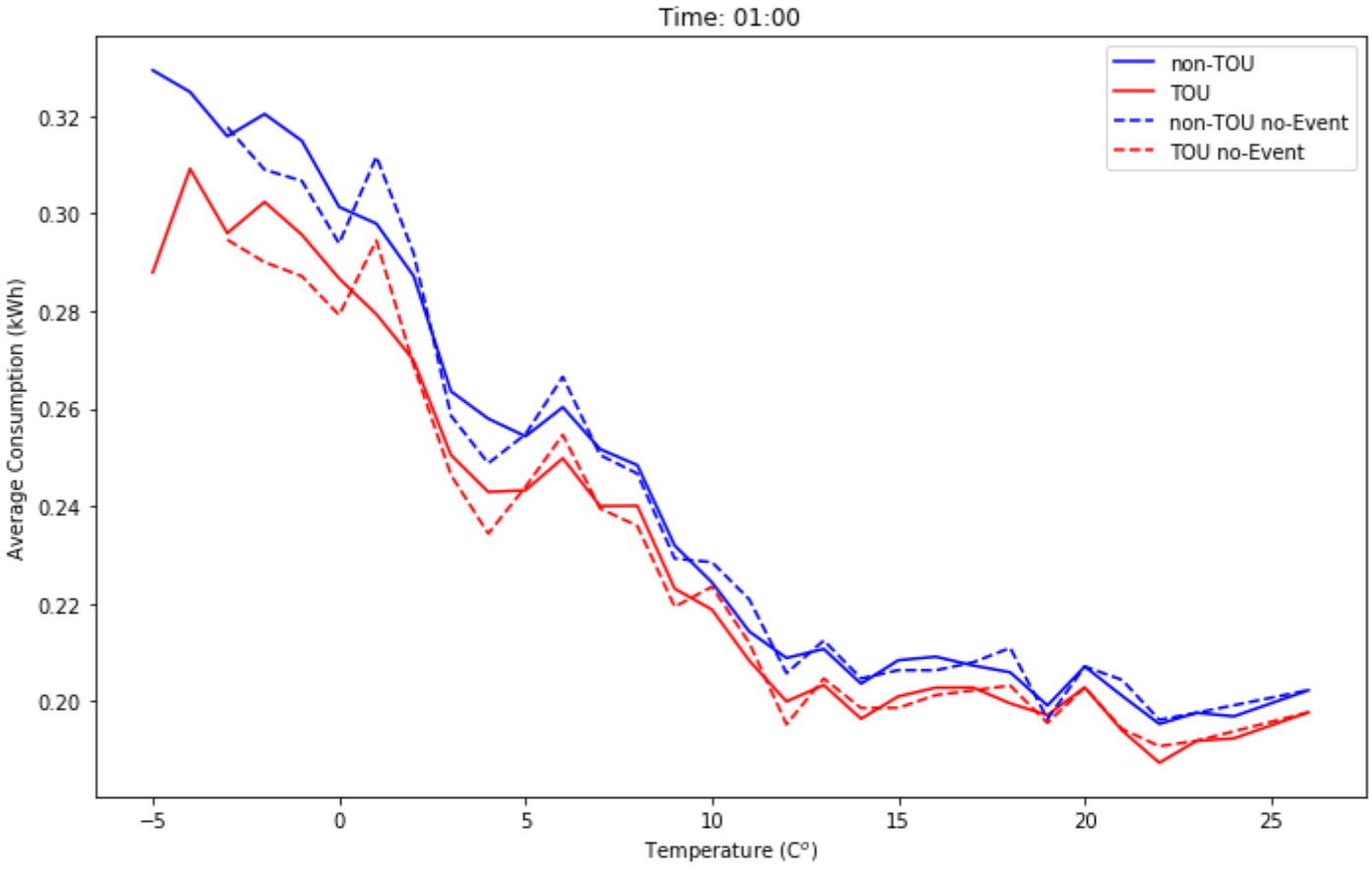

Time: 22:00 (non-TOU)
MSE: 0.00307
MSE: 0.00343
Temperature (C°)

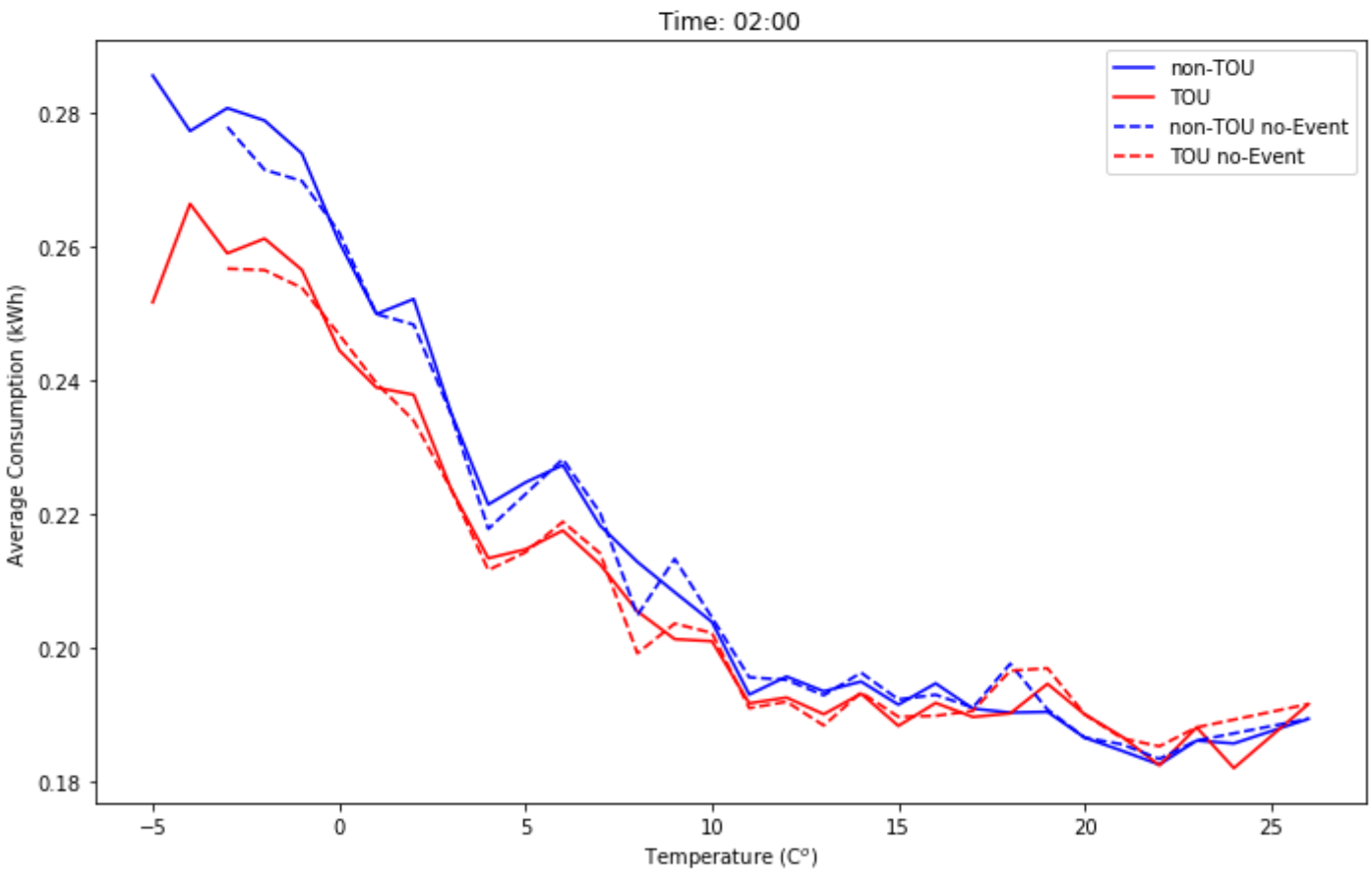

Time: 22:00 (TOU)
MSE: 0.00327
MSE: 0.00365
Temperature (C°)


Average Consum
Temperature (C°)
Average Consum
Temperature (C°)
Time: 23:00 (non-TOU)
MSE: 0.00327
MSE: 0.00353
Average Consumption (kWh)
Temperature (C°)
Time: 23:00 (TOU)
MSE: 0.00382
MSE: 0.00412
Average Consumption (kWh)
Temperature (C°)

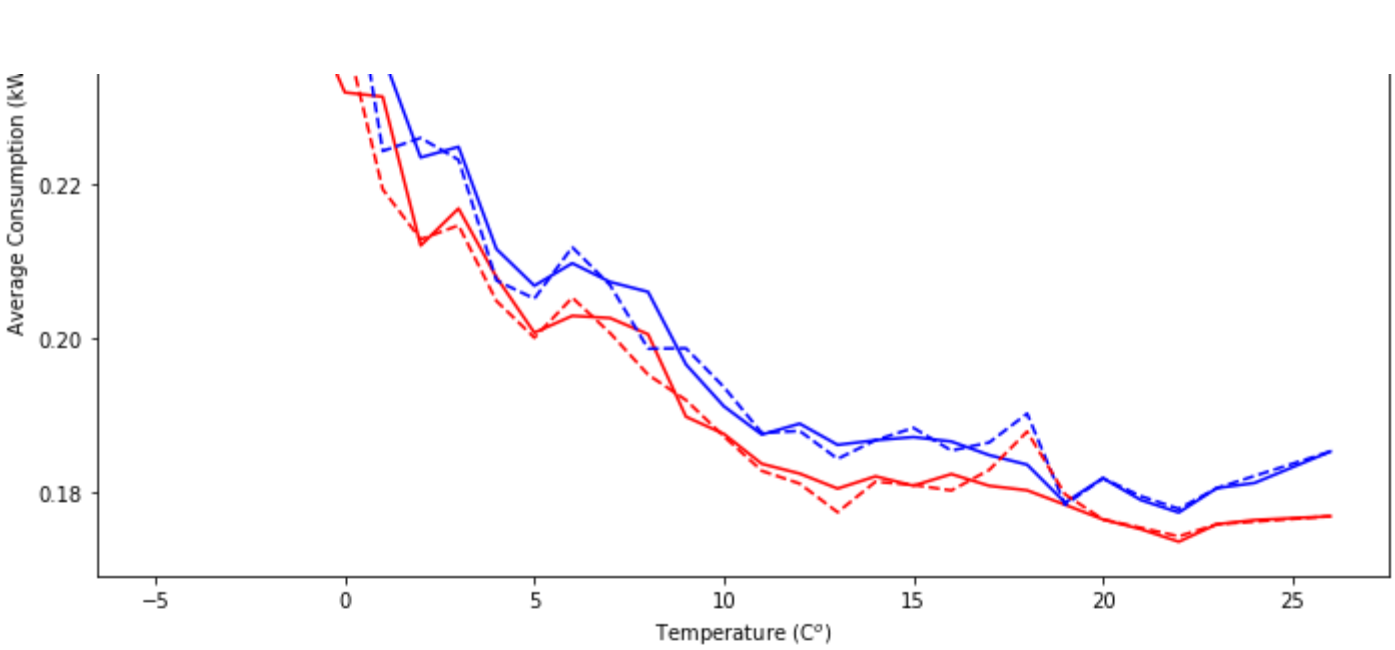
In [78]:
```
# non-TOU vs TOU in terms of average (over both users and the time) hourly energy consumption v.s. temperatur
e in non-Event days
fig_all = plt.figure(figsize = (10,150))
ax_Ntou = [] # store subplot objects
ax_tou = [] # store subplot objects
for i in range(24):
    ax_Ntou.append(fig_all.add_subplot(24, 1, i+1))
    if i <= 9:
        # calculate the average consumption grouped by different temperatures
        df_NdVSt = df_Ntou1h[df_Ntou1h.GMT.str.contains('0' + str(i) + ':00:00')] #non-tou demand vs temperat
ure
        df_NdVSt = df_NdVSt.groupby('TempC')['Total'].mean() / (df_NdVSt.shape[1] - 4)
        df_NdVSt = df_NdVSt.reset_index()
        ax_Ntou[-1].plot(df_NdVSt.TempC, df_NdVSt.Total, c = 'blue', label = 'non-TOU' )

        df_dVSt = df_tou1h[df_tou1h.GMT.str.contains('0' + str(i) + ':00:00')]
        df_dVSt = df_dVSt.groupby('TempC')['Total'].mean() / (df_dVSt.shape[1] - 4)
        df_dVSt = df_dVSt.reset_index()
        ax_Ntou[-1].plot(df_dVSt.TempC, df_dVSt.Total,  c = 'red', label = 'TOU' )

        df_NdVSt = df_Ntou1h_nf[df_Ntou1h_nf.GMT.str.contains('0' + str(i) + ':00:00')] #non-tou demand vs te
mperature for non-Event
        df_NdVSt = df_NdVSt.groupby('TempC')['Total'].mean() / (df_NdVSt.shape[1] - 4)
        df_NdVSt = df_NdVSt.reset_index()
        ax_Ntou[-1].plot(df_NdVSt.TempC, df_NdVSt.Total, c = 'blue', label = 'non-TOU no-Event', linestyle='d
ashed' )

        df_dVSt = df_tou1h_nf[df_tou1h_nf.GMT.str.contains('0' + str(i) + ':00:00')]
        df_dVSt = df_dVSt.groupby('TempC')['Total'].mean() / (df_dVSt.shape[1] - 4)
        df_dVSt = df_dVSt.reset_index()
        ax_Ntou[-1].plot(df_dVSt.TempC, df_dVSt.Total,  c = 'red', label = 'TOU no-Event', linestyle='dashed'
)# TOU demand vs temperature for non-Event

        ax_Ntou[-1].set_title('Time: 0' + str(i) + ':00')
        ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_Ntou[-1].set_ylabel('Average Consumption (kWh)')
        ax_Ntou[-1].legend()
    else:
        df_NdVSt = df_Ntou1h[df_Ntou1h.GMT.str.contains(str(i) + ':00:00')]
        df_NdVSt = df_NdVSt.groupby('TempC')['Total'].mean() / (df_NdVSt.shape[1] - 4)
        df_NdVSt = df_NdVSt.reset_index()
        ax_Ntou[-1].plot(df_NdVSt.TempC, df_NdVSt.Total, c = 'blue', label = 'non-TOU')

        df_dVSt = df_tou1h[df_tou1h.GMT.str.contains(str(i) + ':00:00')]
        df_dVSt = df_dVSt.groupby('TempC')['Total'].mean() / (df_dVSt.shape[1] - 4)
        df_dVSt = df_dVSt.reset_index()
        ax_Ntou[-1].plot(df_dVSt.TempC, df_dVSt.Total,  c = 'red', label = 'TOU' )

        df_NdVSt = df_Ntou1h_nf[df_Ntou1h_nf.GMT.str.contains(str(i) + ':00:00')]
        df_NdVSt = df_NdVSt.groupby('TempC')['Total'].mean() / (df_NdVSt.shape[1] - 4)
        df_NdVSt = df_NdVSt.reset_index()
        ax_Ntou[-1].plot(df_NdVSt.TempC, df_NdVSt.Total, c = 'blue', label = 'non-TOU no-Event', linestyle='d
ashed' ) #non-tou demand vs temperature for non-Event

        df_dVSt = df_tou1h_nf[df_tou1h_nf.GMT.str.contains(str(i) + ':00:00')]
        df_dVSt = df_dVSt.groupby('TempC')['Total'].mean() / (df_dVSt.shape[1] - 4)
        df_dVSt = df_dVSt.reset_index()
        ax_Ntou[-1].plot(df_dVSt.TempC, df_dVSt.Total,  c = 'red', label = 'TOU no-Event', linestyle='dashed'
)# TOU demand vs temperature for non-Event

        ax_Ntou[-1].set_title('Time: ' + str(i) + ':00')
        ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_Ntou[-1].set_ylabel('Average Consumption (kWh)')
        ax_Ntou[-1].legend()
plt.tight_layout()
```

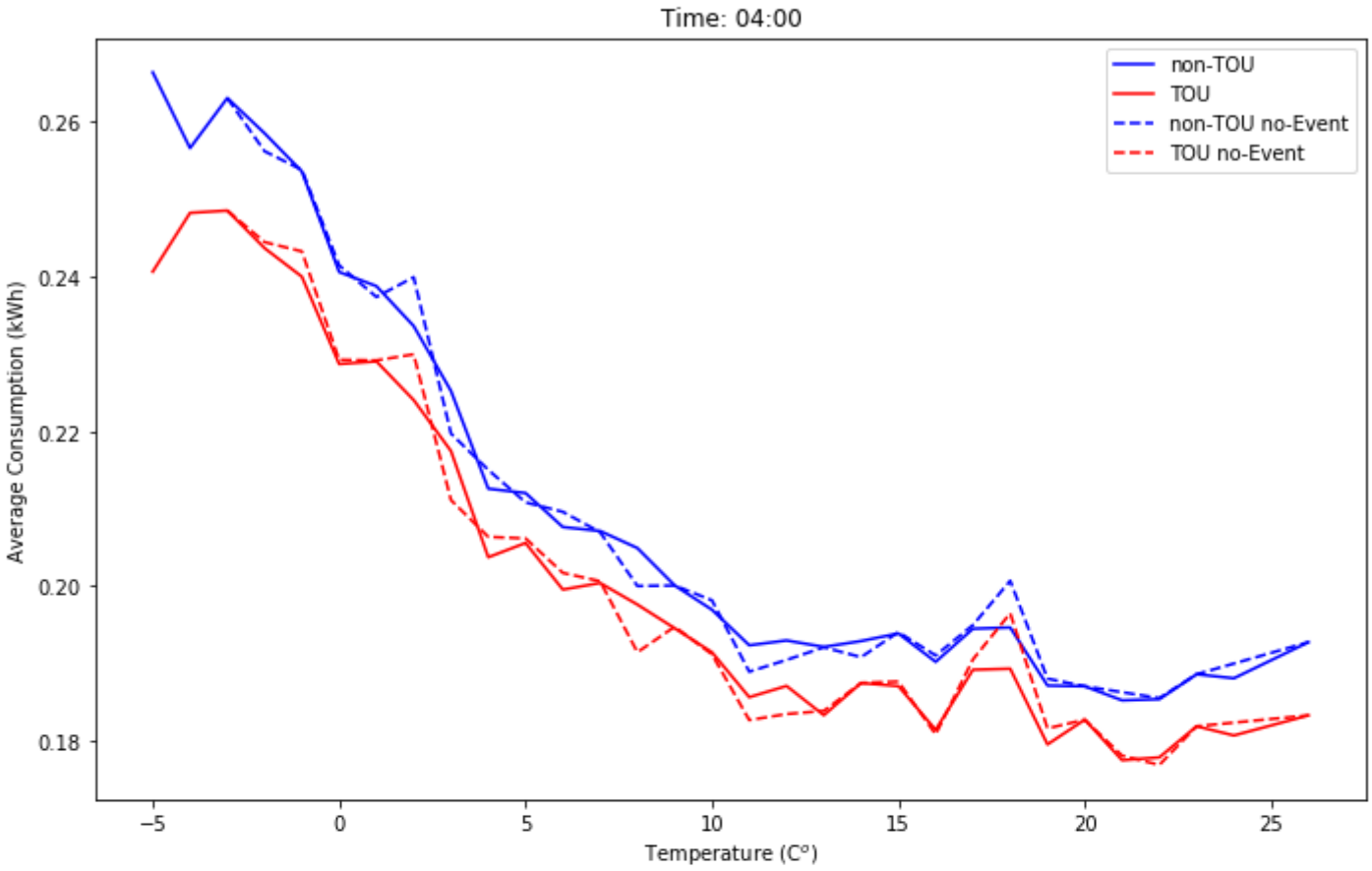

Time: 00:00
non-TOU
TOU
non-TOU no-Event
TOU no-Event
Average Consumption (kWh)
0.40
0.35
0.30
0.25
−5
0
5
10
15
20
25
Temperature (C°)

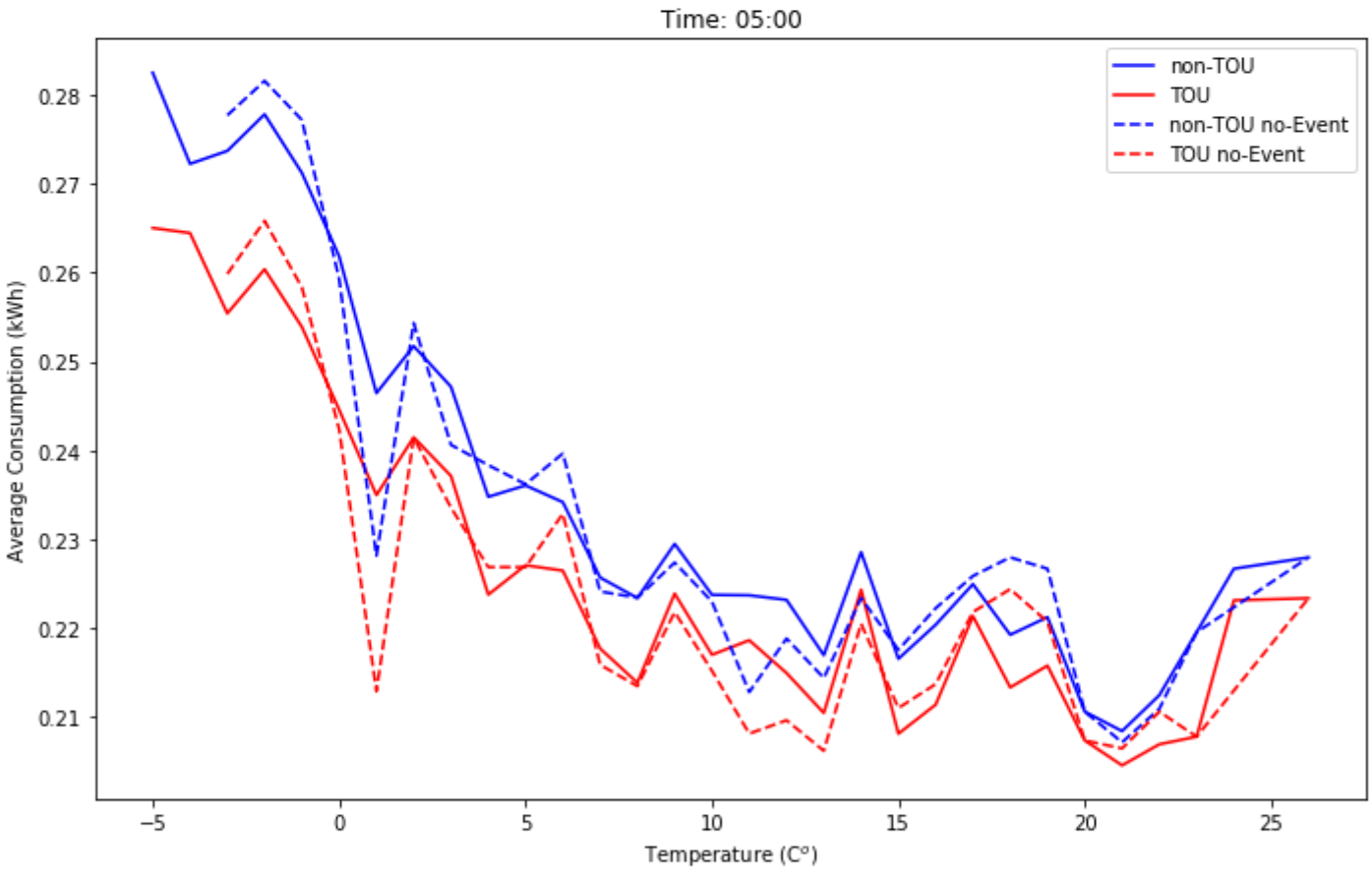

Time: 01:00
non-TOU
TOU
non-TOU no-Event
TOU no-Event
Average Consumption (kWh)
0.32
0.30
0.28
0.26
0.24
0.22
0.20
−5
0
5
10
15
20
25
Temperature (C°)

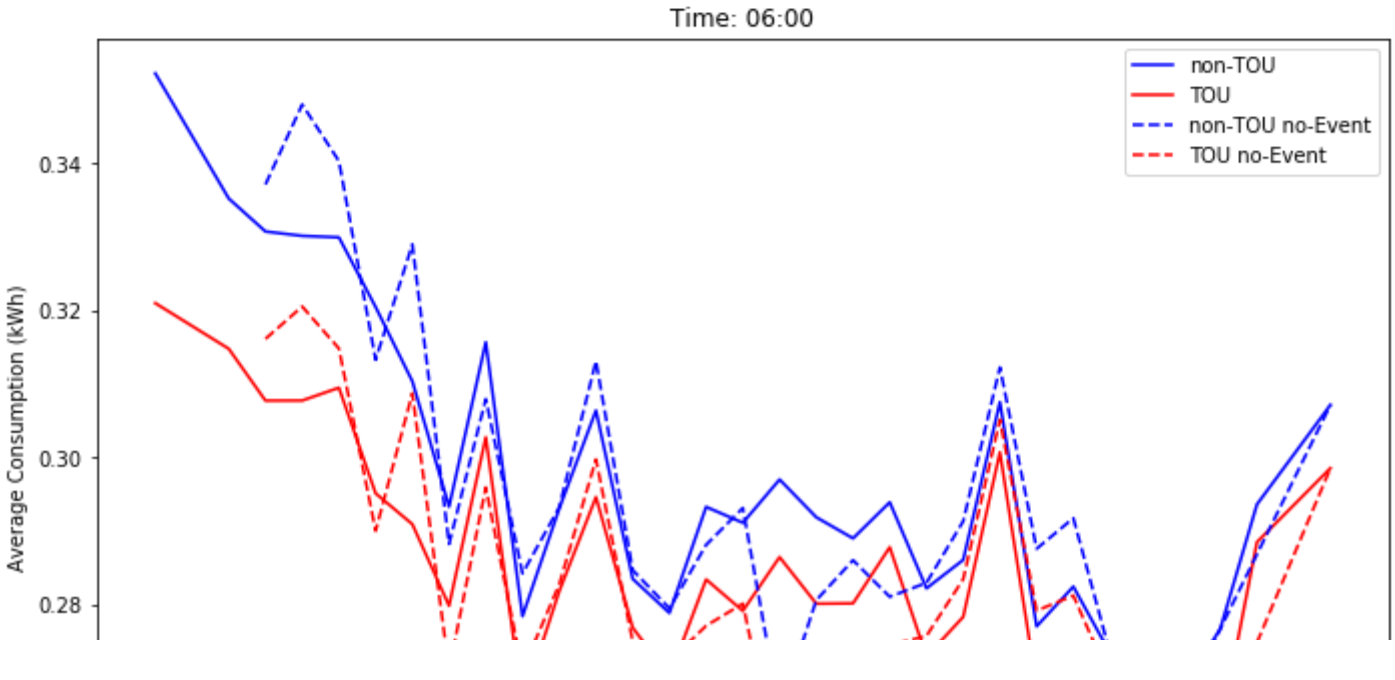

Time: 02:00
non-TOU
TOU
non-TOU no-Event
TOU no-Event
Average Consumption (kWh)
0.28
0.26
0.24
0.22
0.20
0.18
−5
0
5
10
15
20
25
Temperature (C°)

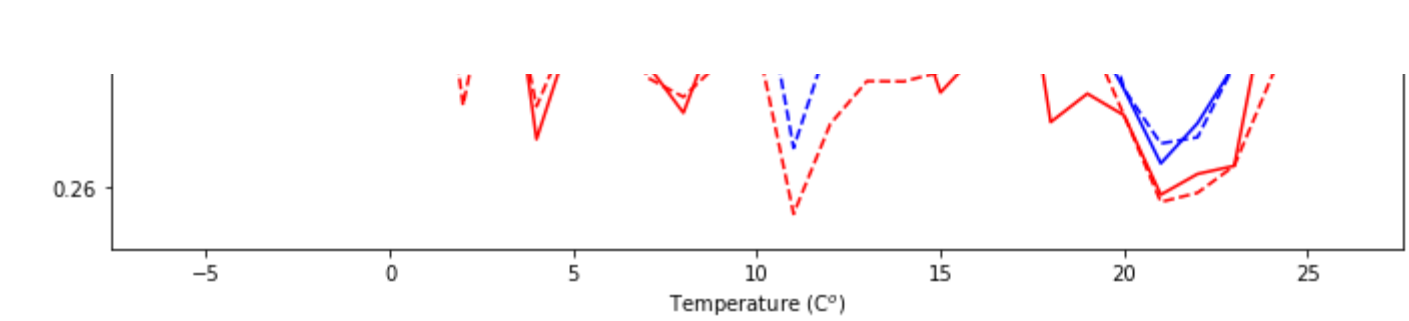

Time: 03:00
non-TOU
TOU
non-TOU no-Event
TOU no-Event
0.26
0.24

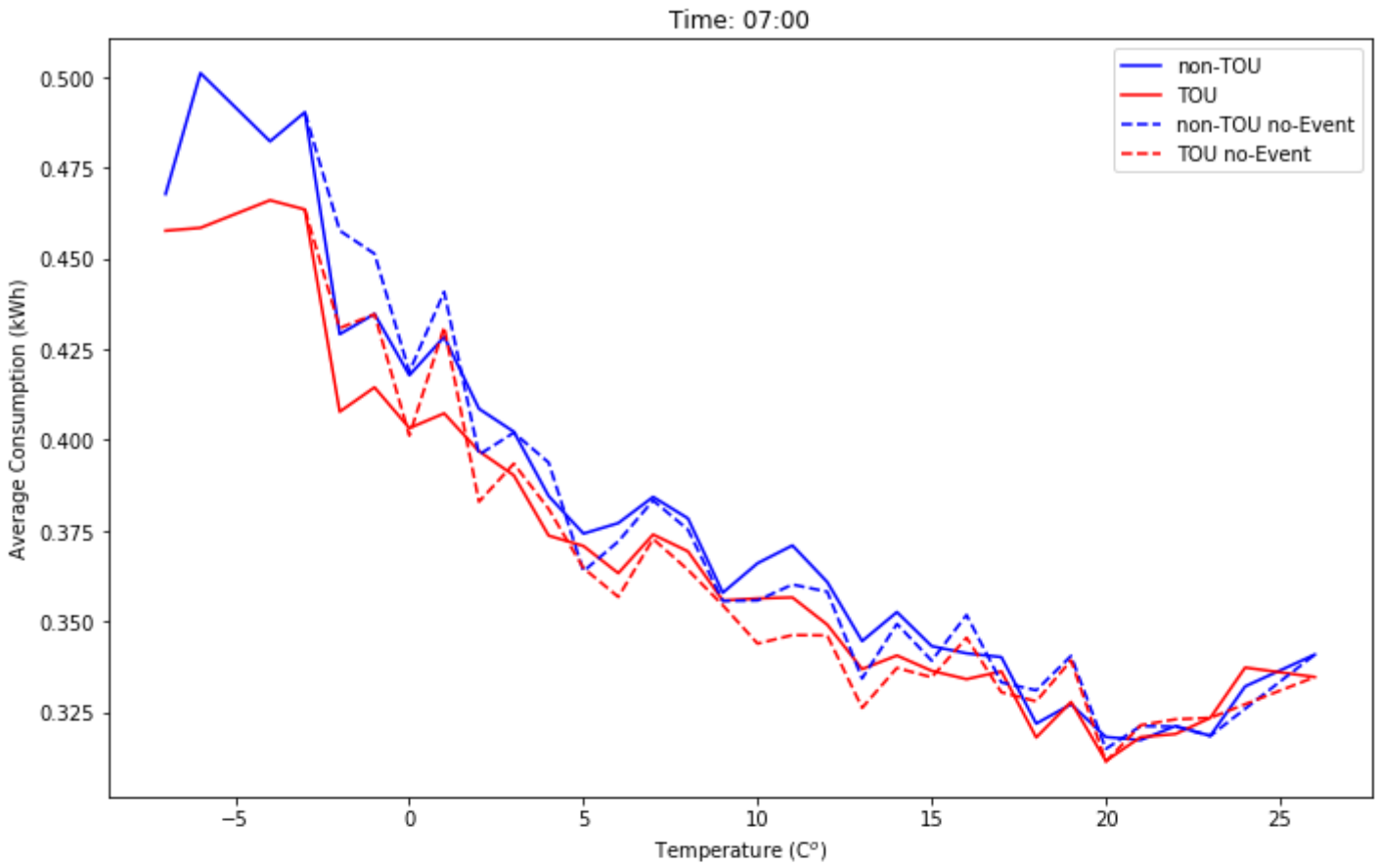

Average Consumption (kW
0.22
0.20
0.18
−5
0
5
10
15
20
25
Temperature (C°)

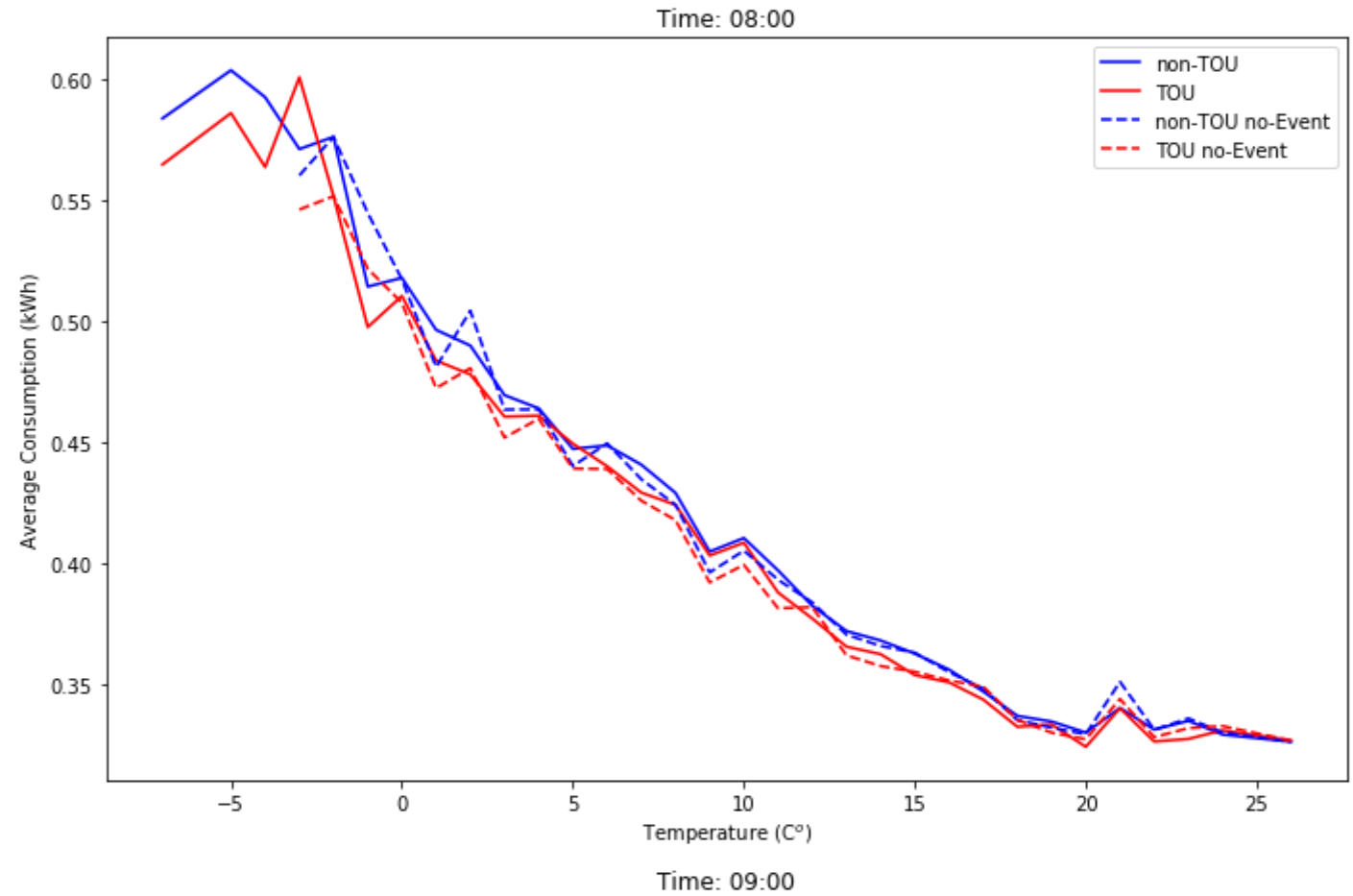

Time: 04:00
non-TOU
TOU
non-TOU no-Event
TOU no-Event
Average Consumption (kWh)
0.26
0.24
0.22
0.20
0.18
−5
0
5
10
15
20
25
Temperature (C°)

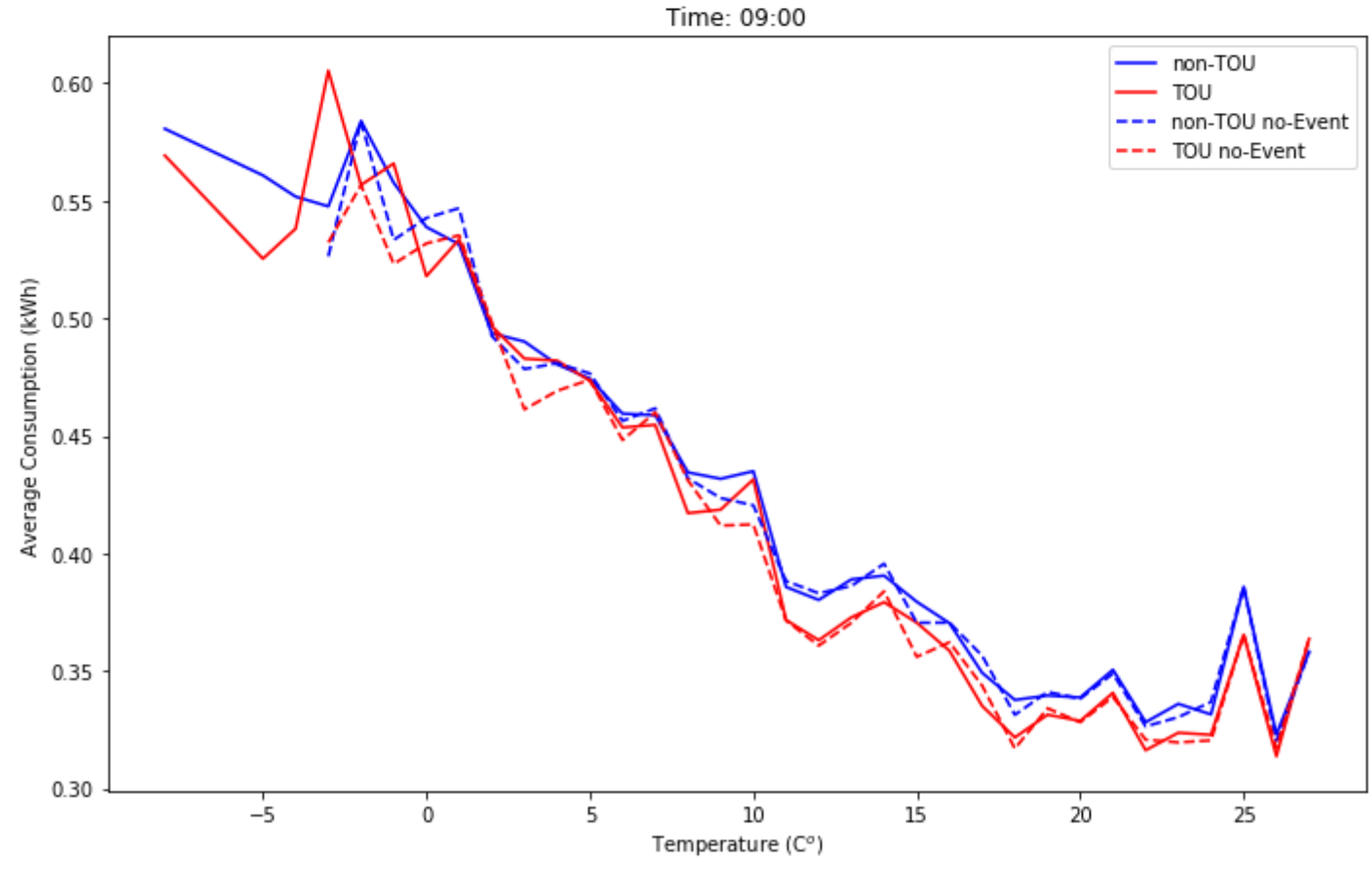

Time: 05:00
non-TOU
TOU
non-TOU no-Event
TOU no-Event
Average Consumption (kWh)
0.28
0.27
0.26
0.25
0.24
0.23
0.22
0.21
−5
0
5
10
15
20
25
Temperature (C°)


Time: 06:00
non-TOU
TOU
non-TOU no-Event
TOU no-Event
Average Consumption (kWh)
0.34
0.32
0.30
0.28

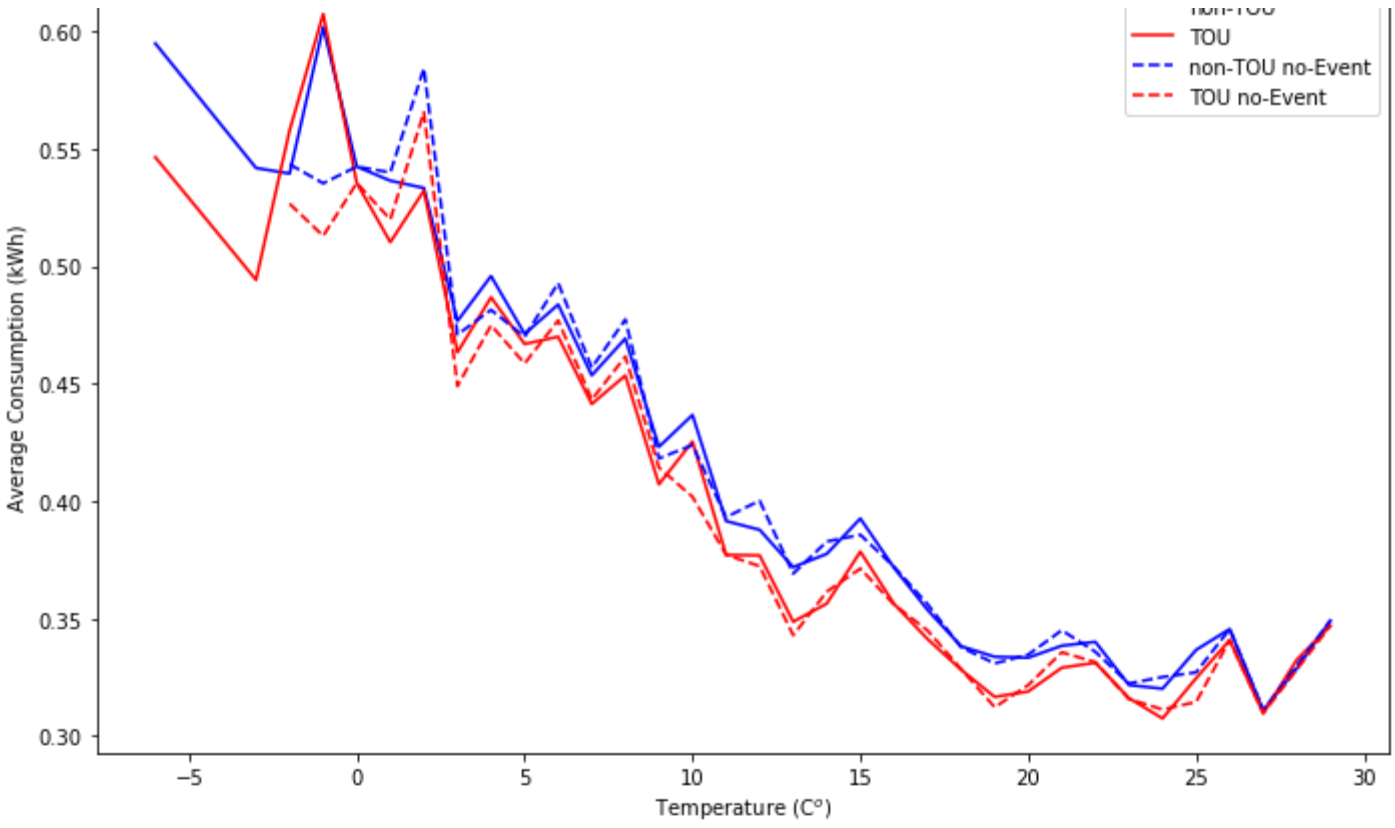

0.26
−5
0
5
10
15
20
25
Temperature (C°)


Time: 07:00
Average Consumption (kWh)
0.500
0.475
0.450
0.425
0.400
0.375
0.350
0.325
non-TOU
TOU
non-TOU no-Event
TOU no-Event
−5
0
5
10
15
20
25
Temperature (C°)

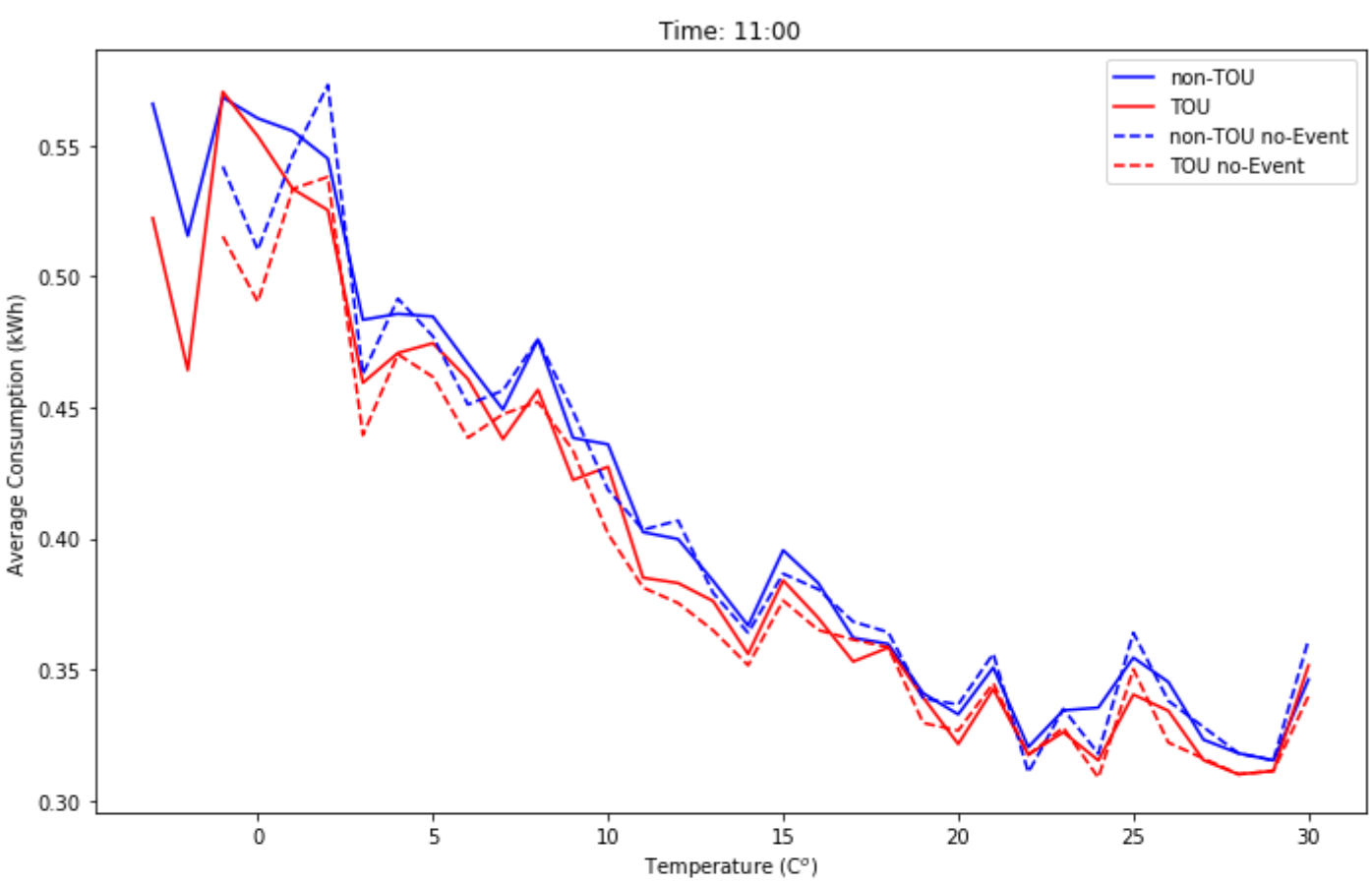

Time: 08:00
Average Consumption (kWh)
0.60
0.55
0.50
0.45
0.40
0.35
non-TOU
TOU
non-TOU no-Event
TOU no-Event
−5
0
5
10
15
20
25
Temperature (C°)


Time: 09:00
Average Consumption (kWh)
0.60
0.55
0.50
0.45
0.40
0.35
0.30
non-TOU
TOU
non-TOU no-Event
TOU no-Event
−5
0
5
10
15
20
25
Temperature (C°)

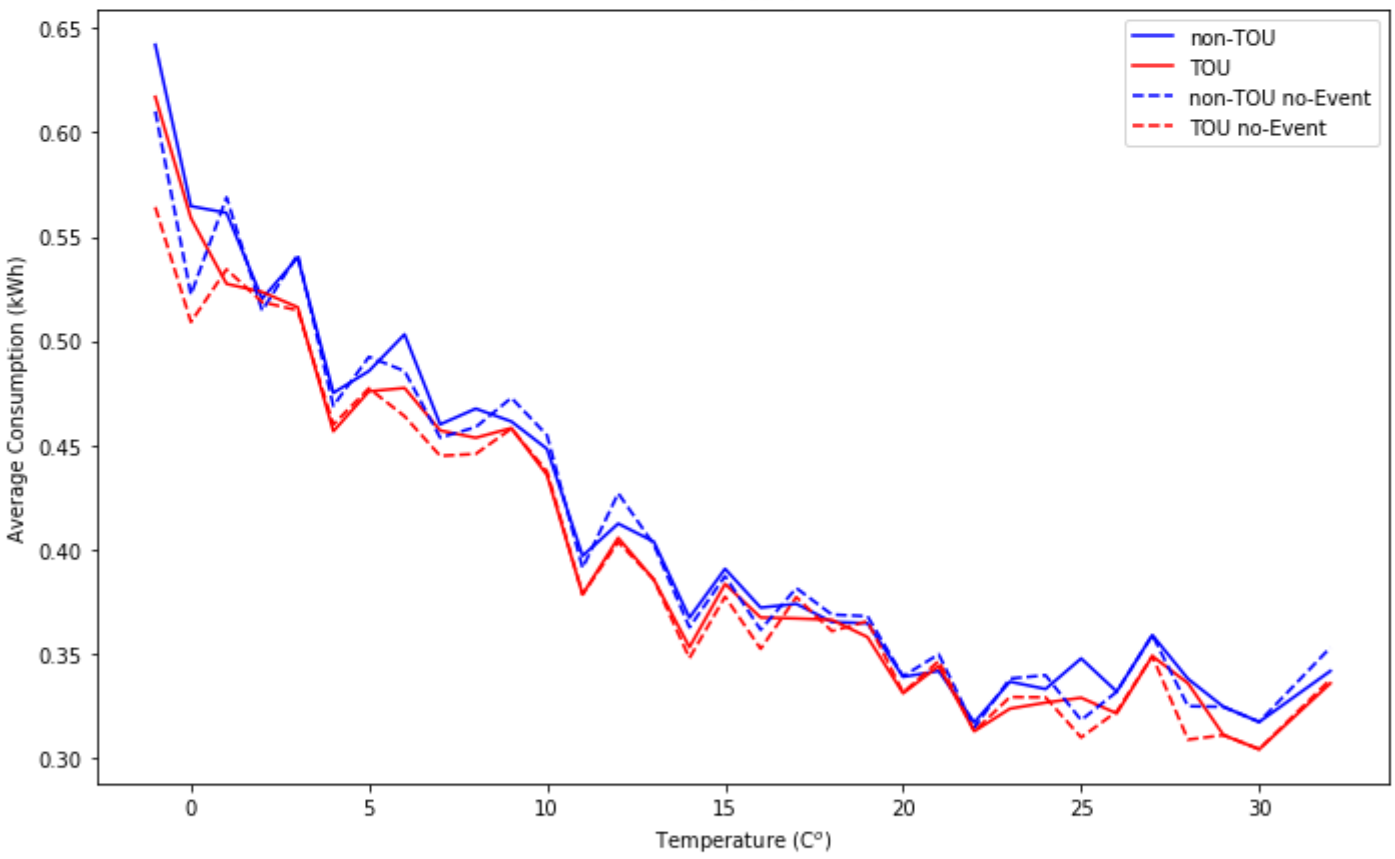
Time: 10:00



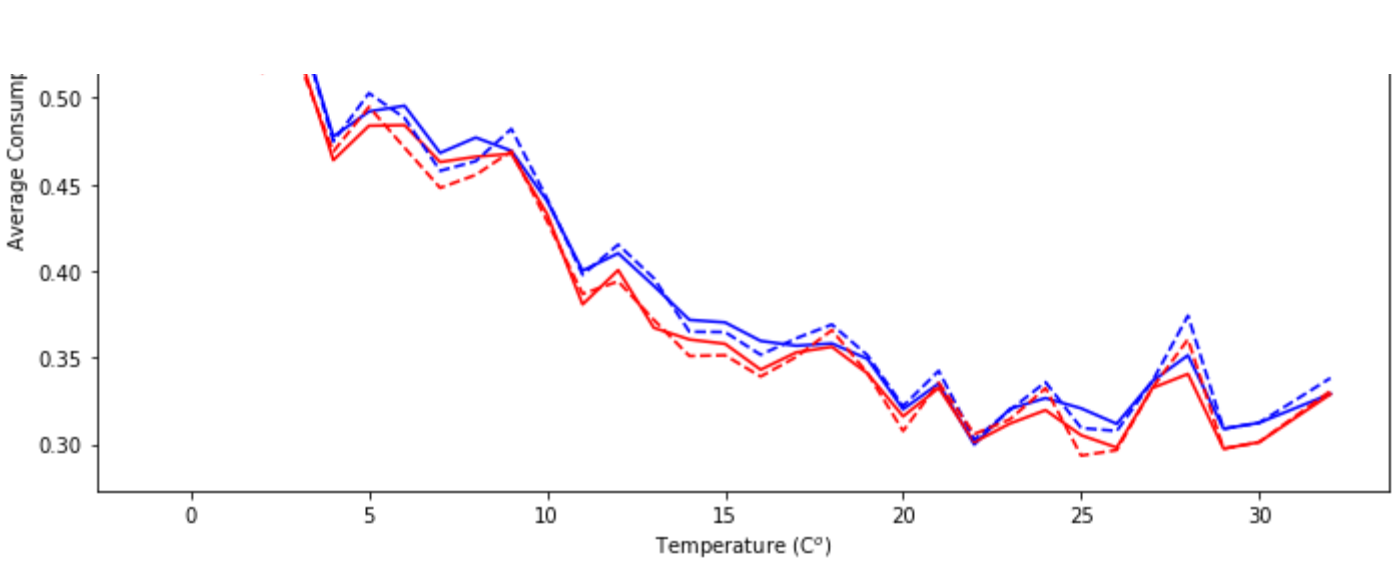
Time: 11:00

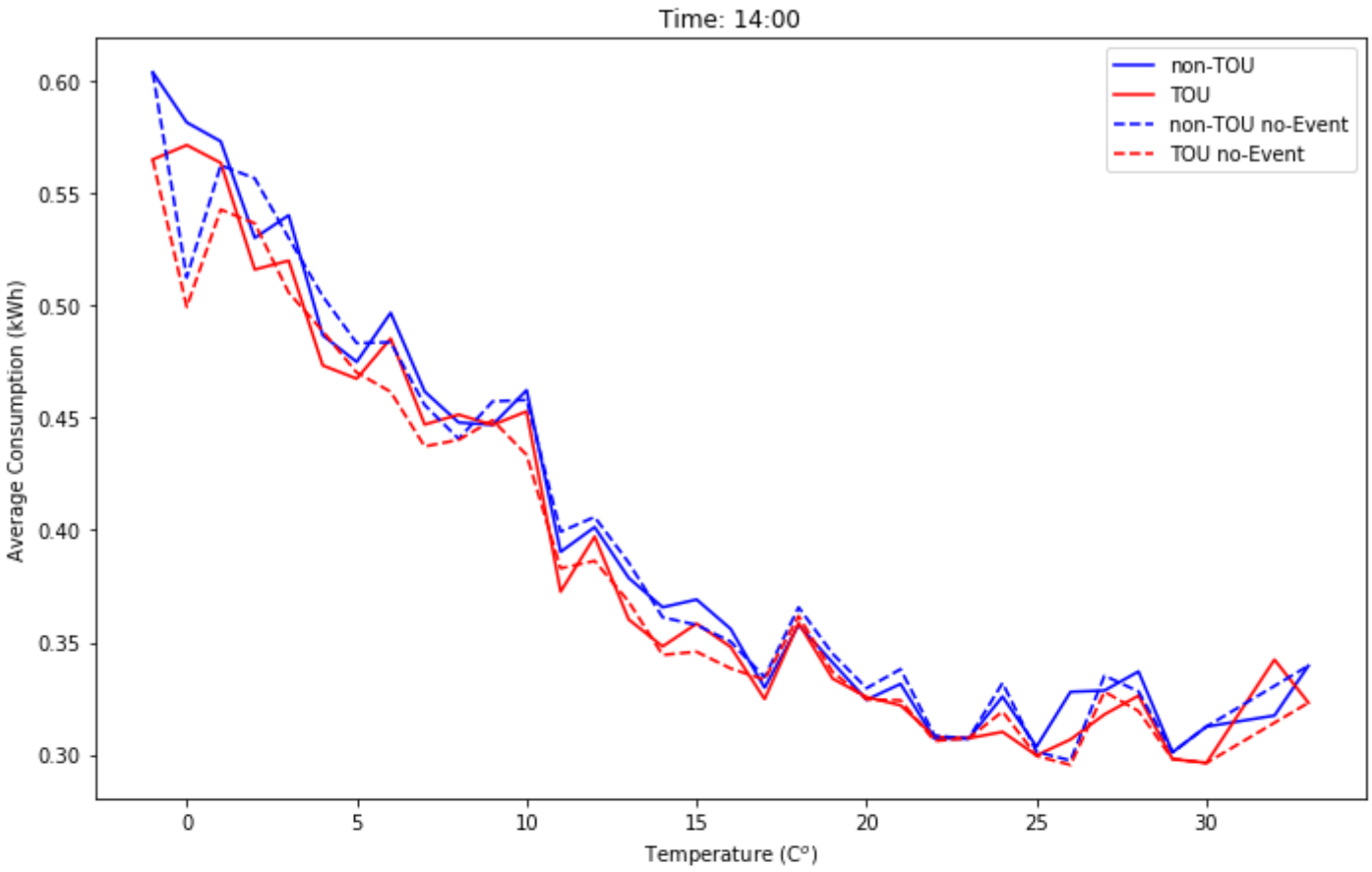


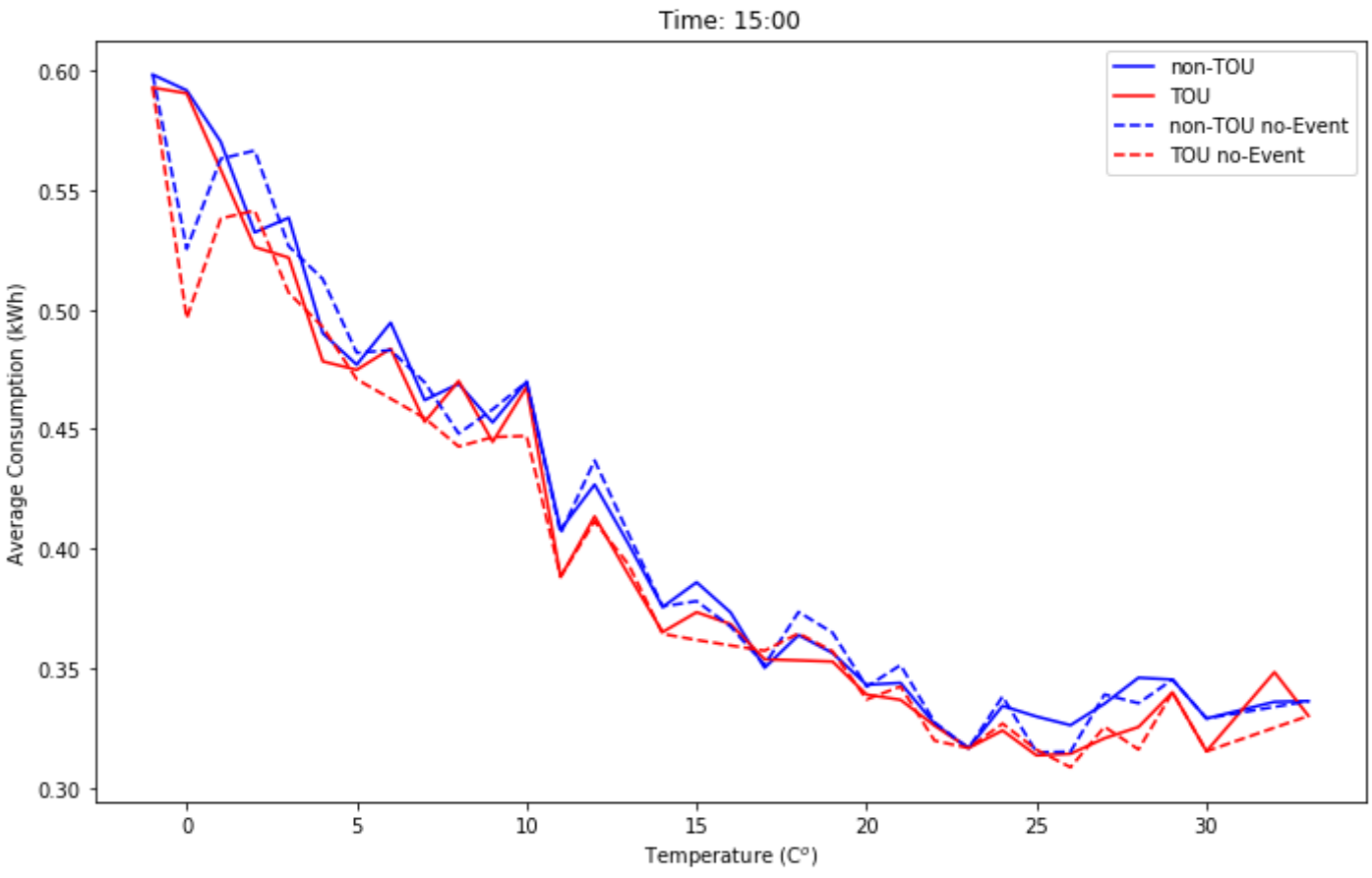
Time: 12:00

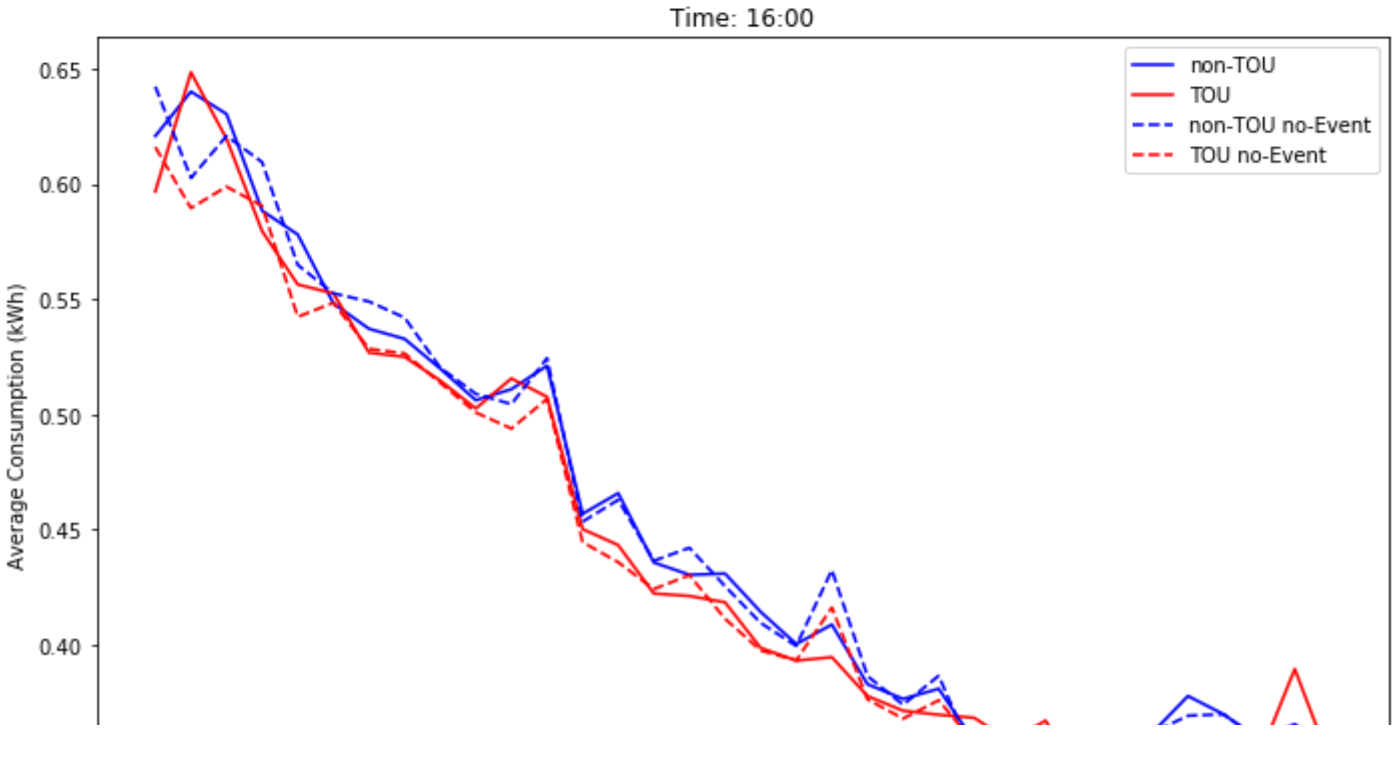


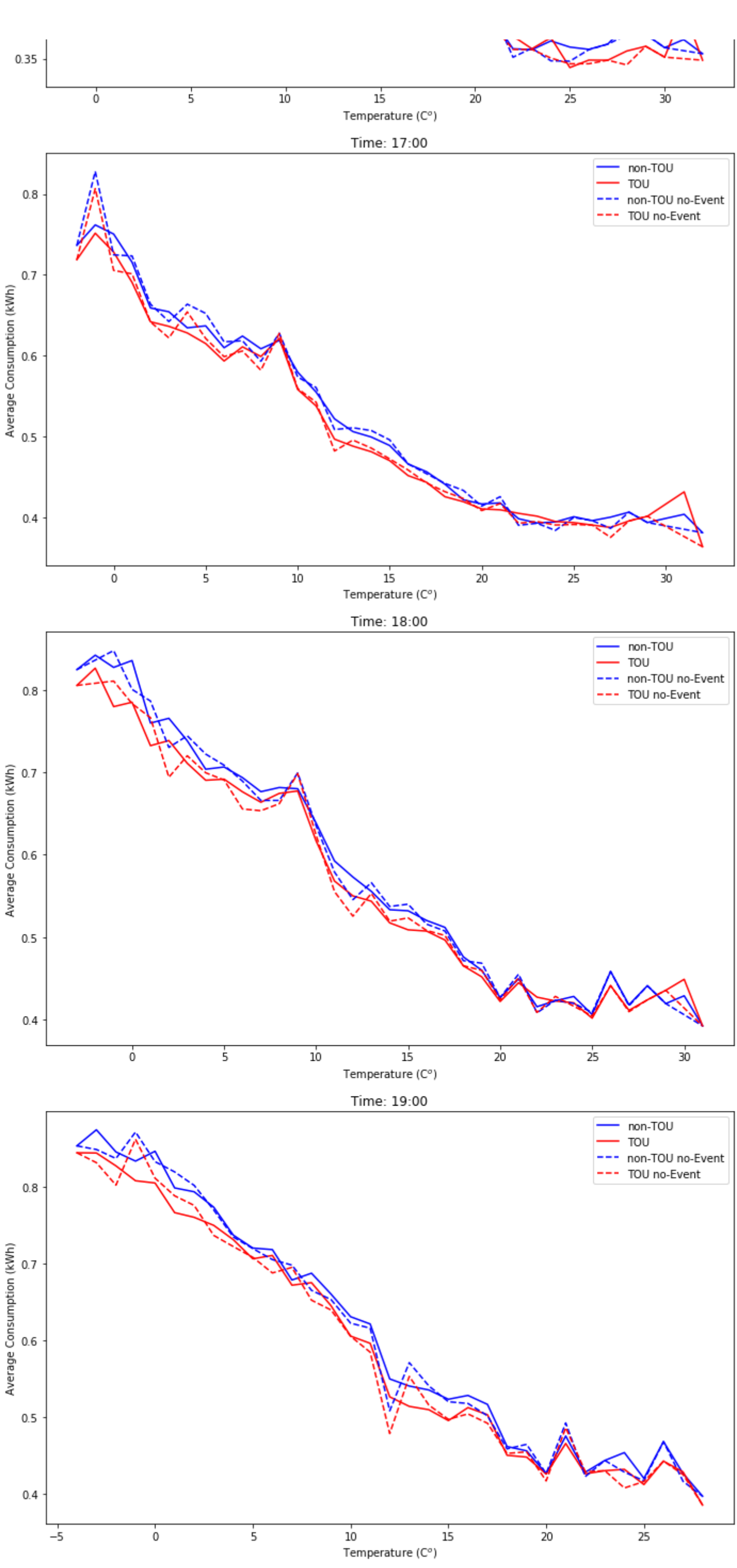
Time: 13:00

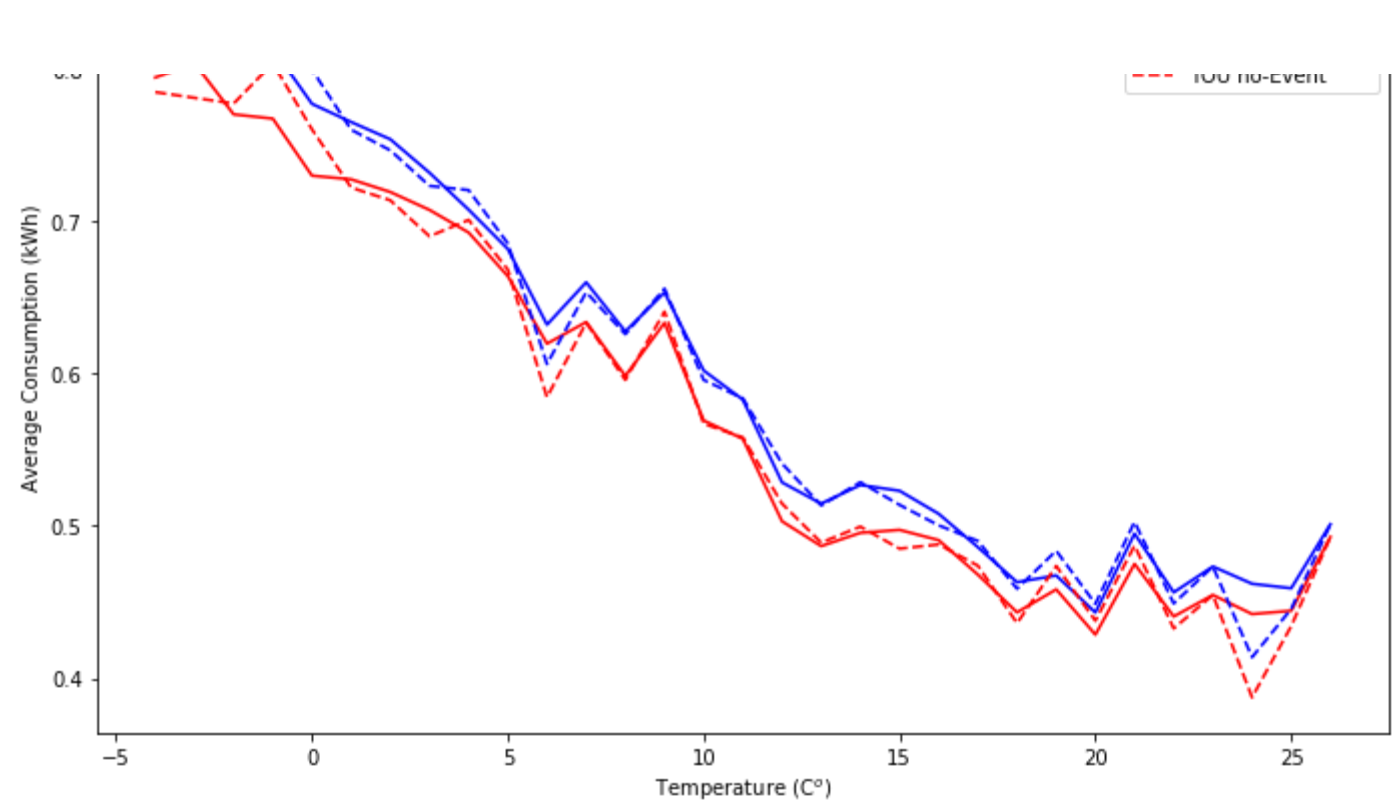

Average Consump
0.50
0.45
0.40
0.35
0.30
0
5
10
15
20
25
30
Temperature (C°)


Time: 14:00
Average Consumption (kWh)
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0
5
10
15
20
25
30
Temperature (C°)
non-TOU
TOU
non-TOU no-Event
TOU no-Event

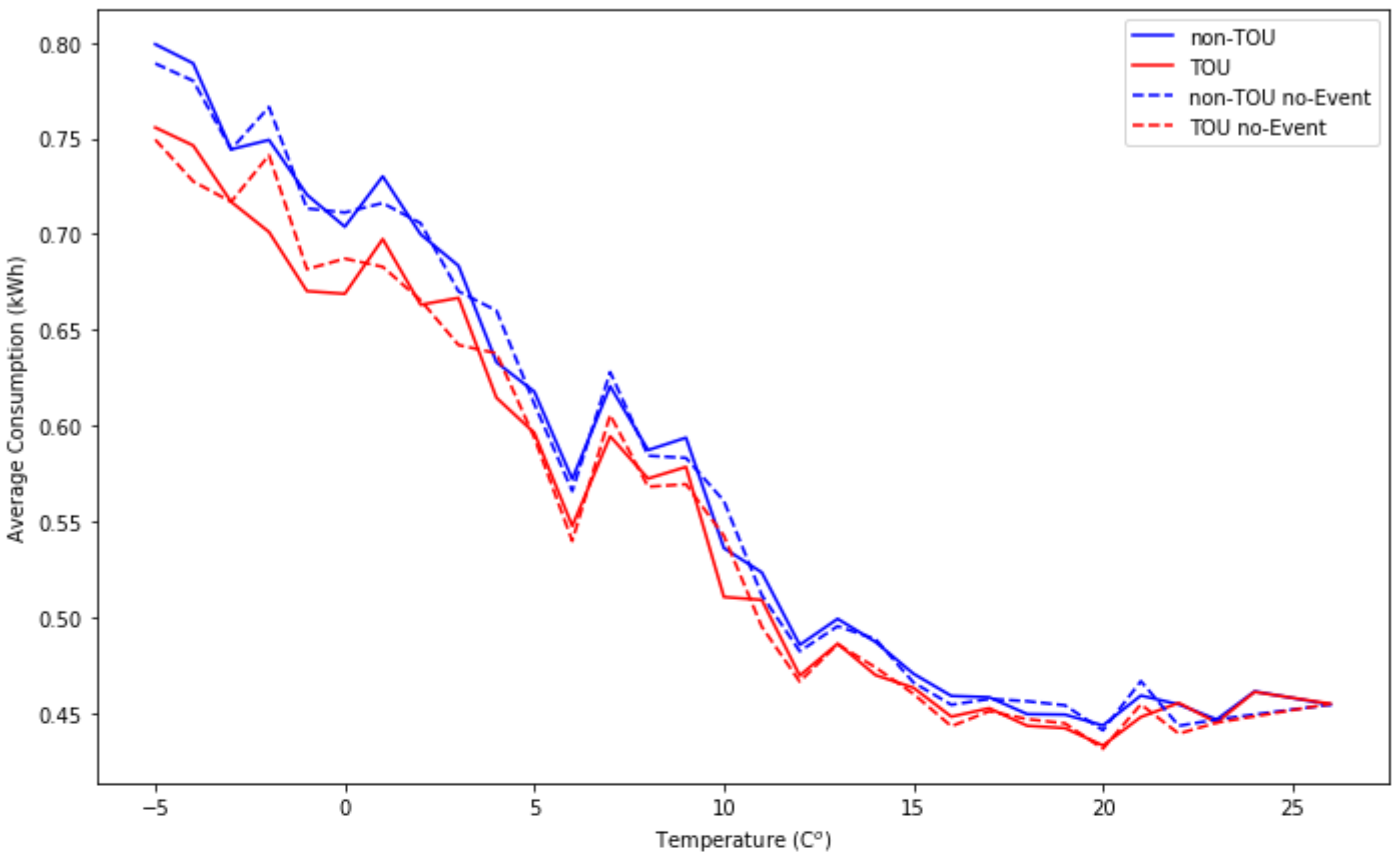

Time: 15:00
Average Consumption (kWh)
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0
5
10
15
20
25
30
Temperature (C°)
non-TOU
TOU
non-TOU no-Event
TOU no-Event


Time: 16:00
Average Consumption (kWh)
0.65
0.60
0.55
0.50
0.45
0.40
non-TOU
TOU
non-TOU no-Event
TOU no-Event

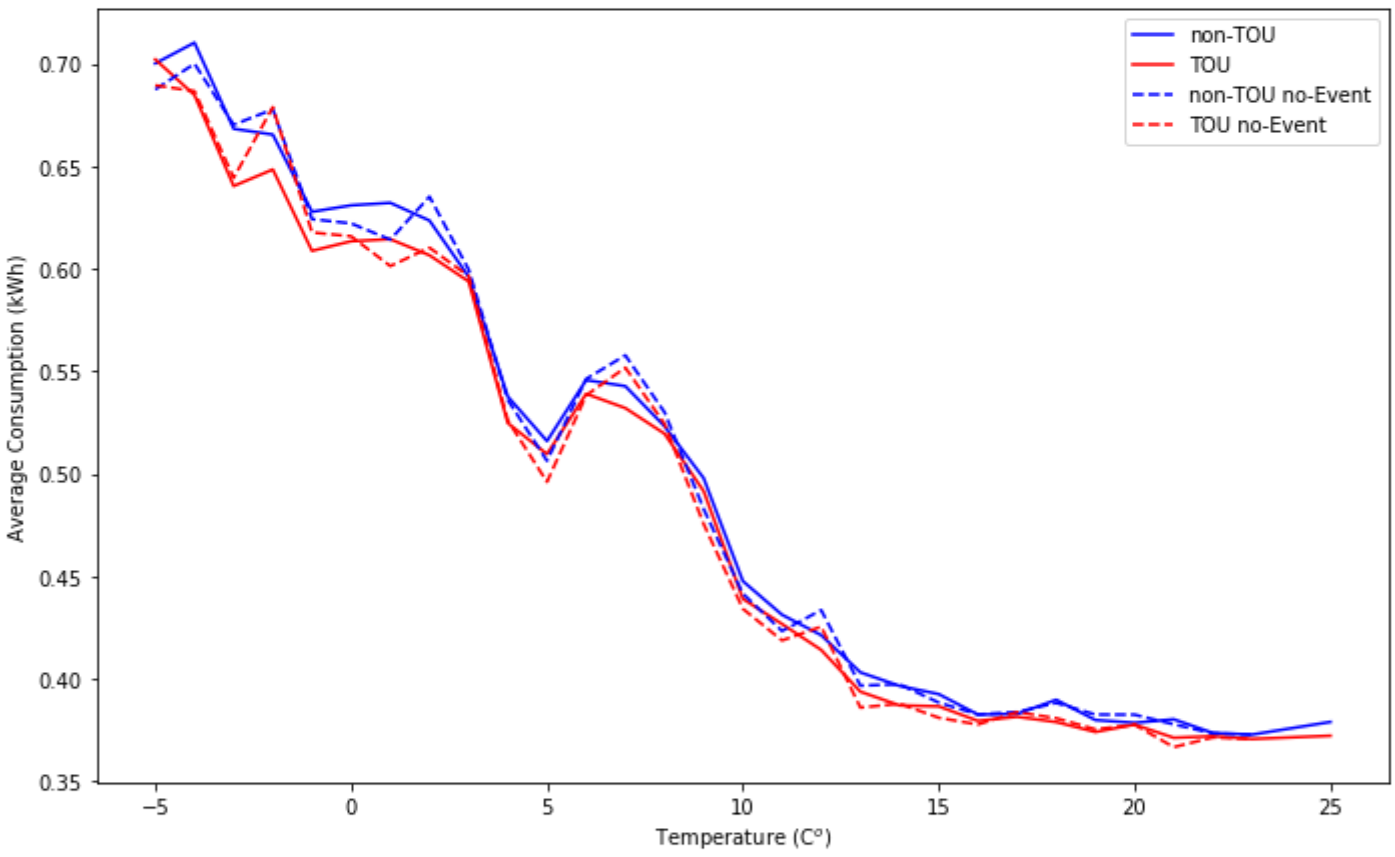

0.35
0
5
10
15
20
25
30
Temperature (C°)
Time: 17:00
non-TOU
TOU
non-TOU no-Event
TOU no-Event
Average Consumption (kWh)
0.8
0.7
0.6
0.5
0.4
Temperature (C°)
Time: 18:00
Time: 19:00
Time: 20:00
−5



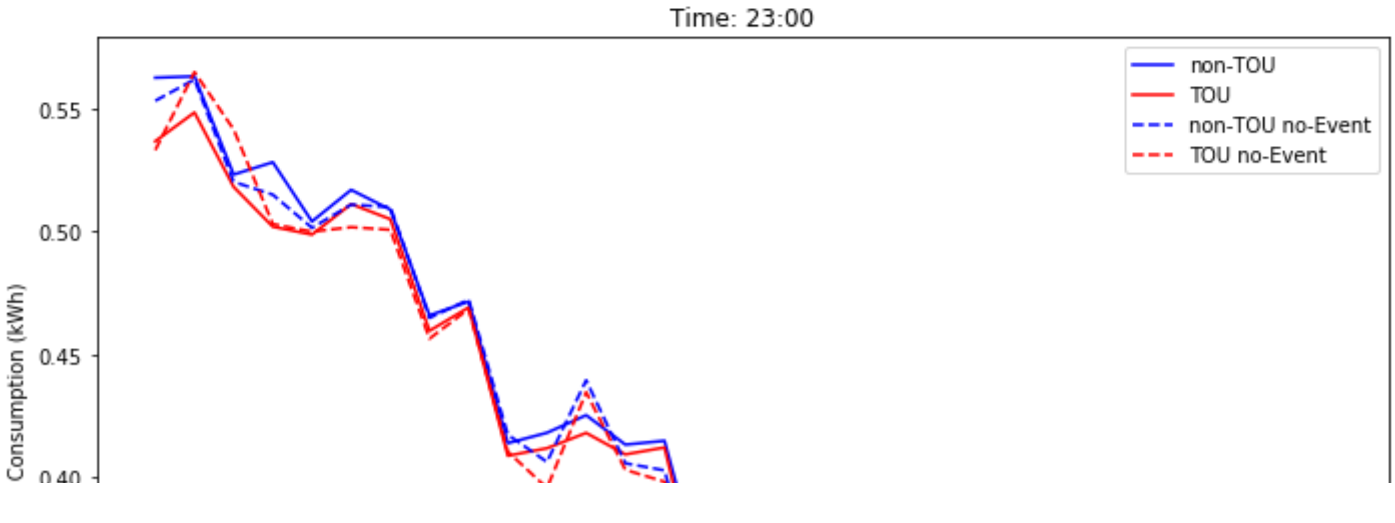
Time: 21:00

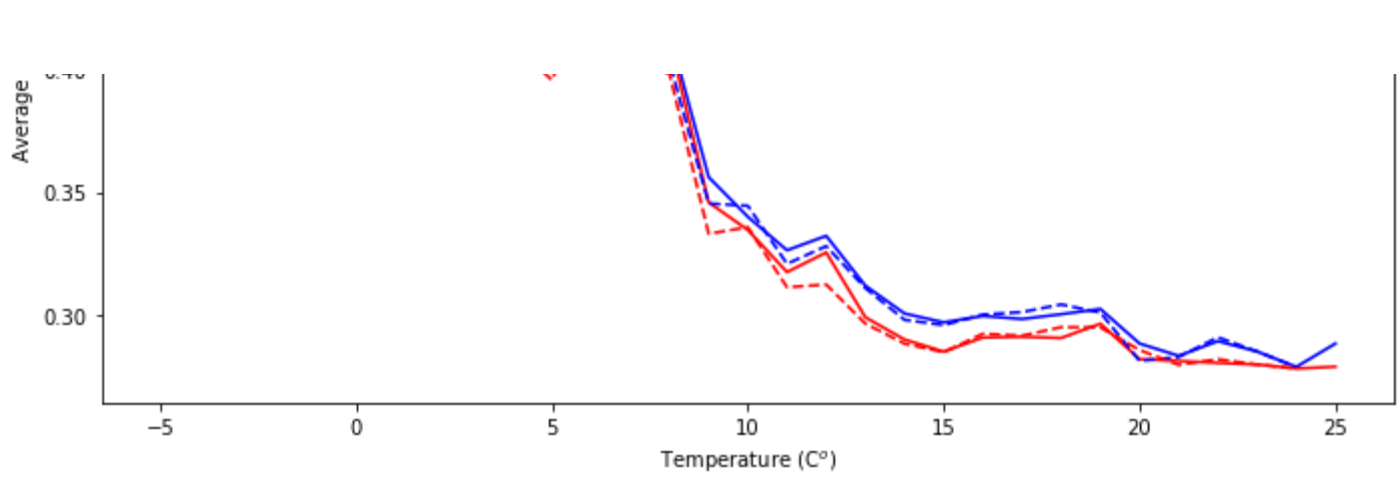


Time: 22:00



Time: 23:00





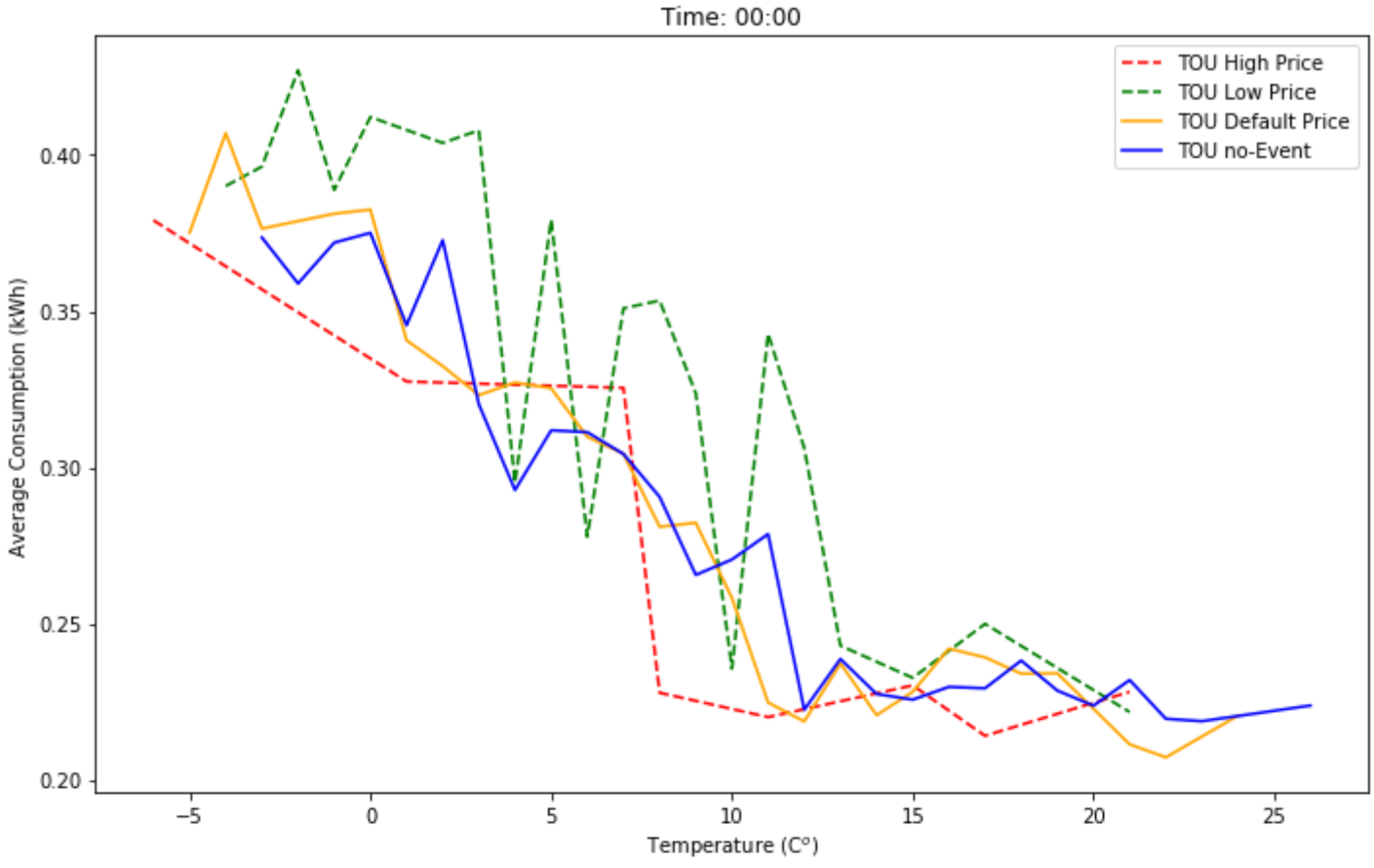
When there's no event, the fluctuation patterns of TOU (i.e., default price 0.1176 in this case) and non-TOU (0.14228) are quite similar, except for a gap. It's worth to analyze the reasons of the gap. Since we have removed effects of price fluctuation, both are fixed prices in TOU and non-TOU, the only difference could be 1) different price values, 0.1176 (TOU) vs 0.14228 (non-TOU); 2) different appliance usage characteristic between TOU group and non-TOU group; 3) others(like attitude, incomes etc.) And the later two reasons seems more reasonable, since non-TOU price 0.14228 > 0.1176 (TOU), but the consumption of non-TOU more than the TOU case, which is counter intuitive. If two groups have simliar preference, then they would consume more when price is lower.

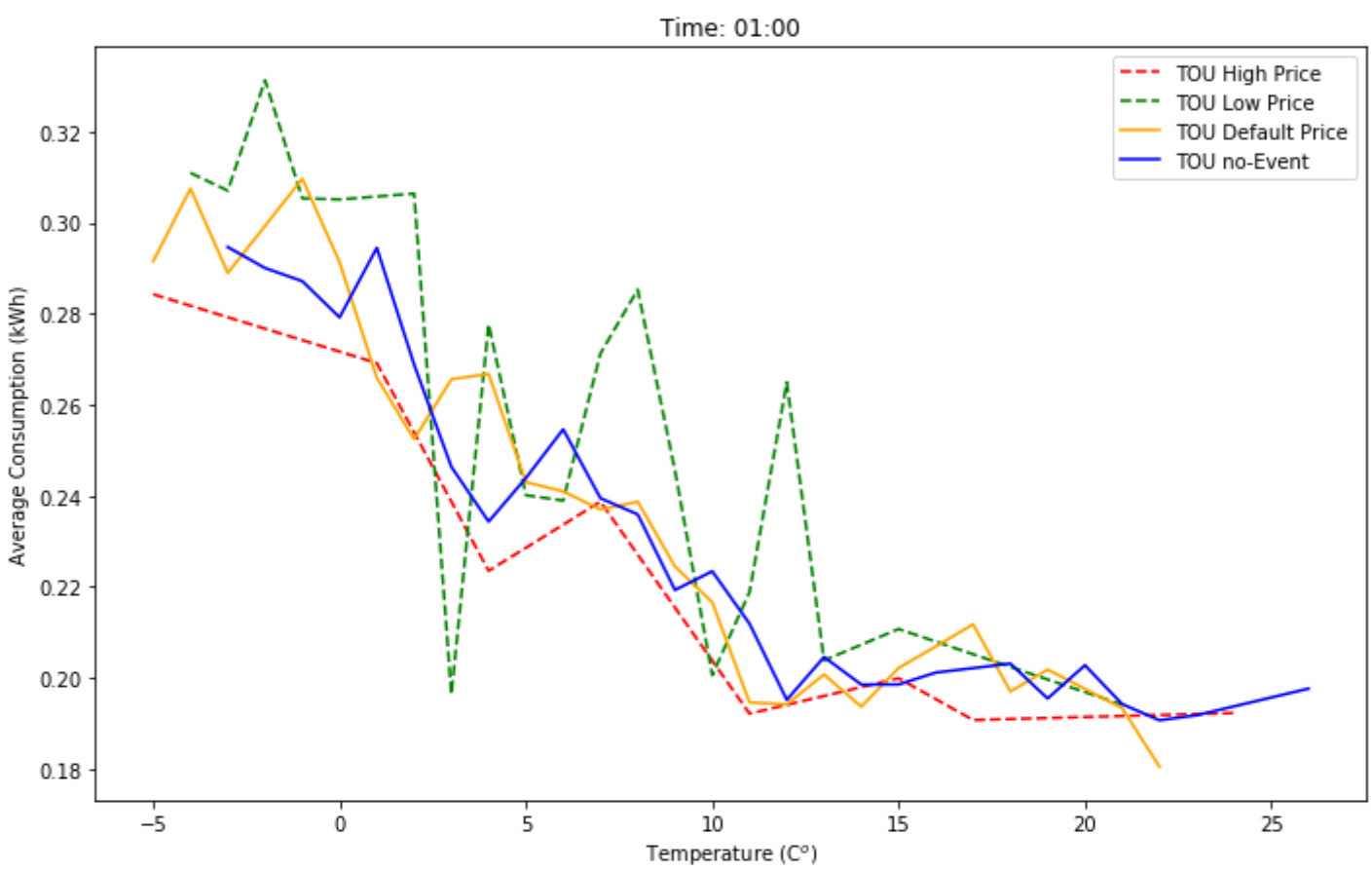
The similar patterns give us some evidence that our approach of removing event days (removing one day seems enough) kind of filters out effects of price changes to the consumption in TOU case. Thus, we can now compare event days' consumption when pricing changing happens with the non-Event TOU consumption.

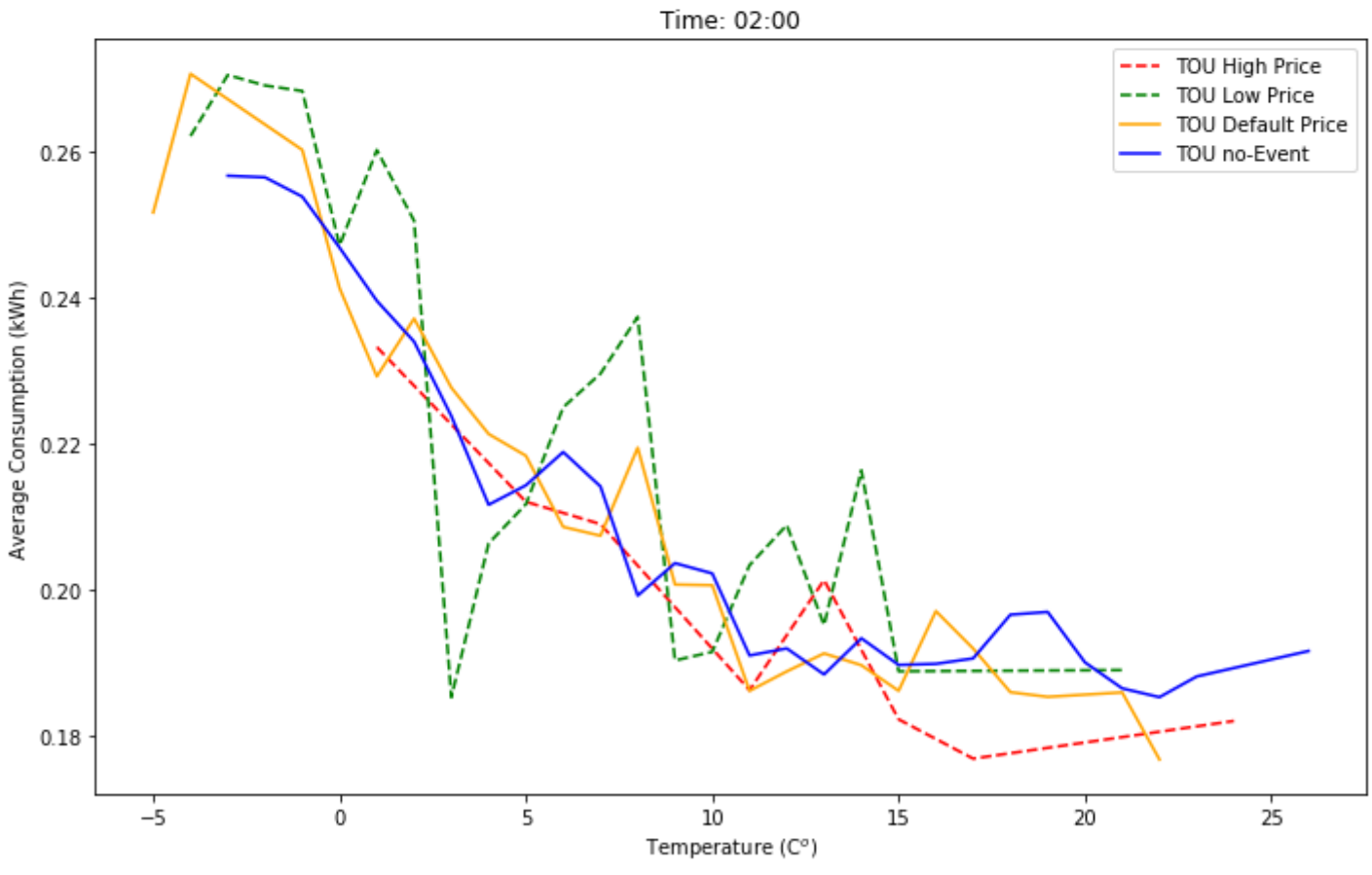
Now let's compare the average TOU consumption in event periods, i.e., high-price or low-price periods with the no-event periods, and see how the price change affects the consumption.

In [68]:
```
# High price periods high 0.672 vs. default 0.1176 and low 0.0399
# Note: CM events only use high and low prices, so when we pick the high price and low price periods,
# those in CM will also be picked.
# first, create a list of days that belongs to event days
h_event_time = set()
df_event = df_tariff_1h[df_tariff_1h.Event_tags.notnull()]
# df_Ntou1h_hPrice = df_Ntou1h[df_tariff_1h.Price == 0.672]
# df_Ntou1h_lPrice = df_Ntou1h[df_tariff_1h.Price == 0.0399]
df_tou1h_hPrice = df_tou1h[df_tariff_1h.Price == 0.672]
df_tou1h_lPrice = df_tou1h[df_tariff_1h.Price == 0.0399]

# All event day's dataframe
df_help_event = df_tariff_1h[df_help['GMT'].isin(event_days)]
# the dataframe in event days with default price
df_help_event = df_help_event[df_help_event.Price == 0.1176]
# TOU dataframe corresponding to these default price in event days
df_tou1h_eventday_dPrice = df_tou1h.loc[df_help_event.index]
```

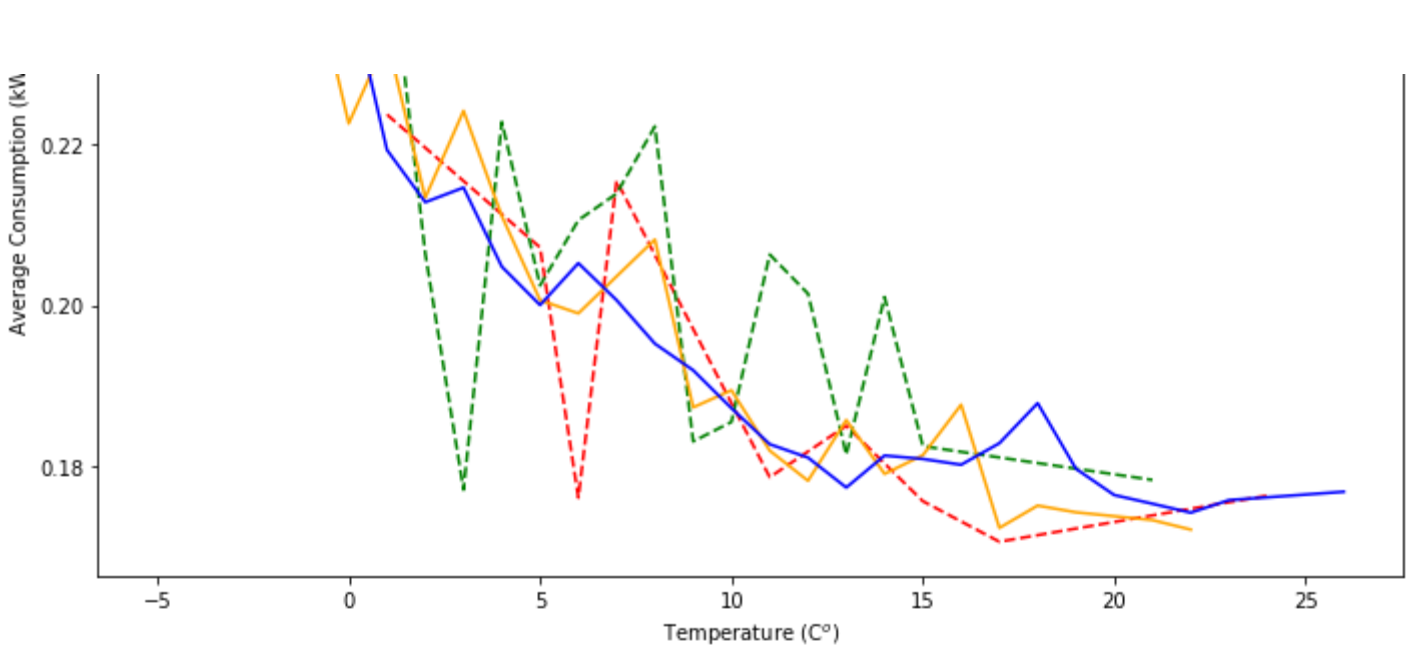
In [96]:
```
# non-TOU vs TOU in terms of average (over both users and the time) hourly energy consumption v.s. temperatur
e
fig_all = plt.figure(figsize = (10,150))
ax_Ntou = [] # store subplot objects
ax_tou = [] # store subplot objects
for i in range(24):
    ax_Ntou.append(fig_all.add_subplot(24, 1, i+1))
    if i <= 9:
        # calculate the average consumption grouped by different temperatures
        df_dVSt = df_tou1h_hPrice[df_tou1h_hPrice.GMT.str.contains('0' + str(i) + ':00:00')] #TOU demand vs t
emperature for high price periods
        df_dVSt = df_dVSt.groupby('TempC')['Total'].mean() / (df_dVSt.shape[1] - 4)
        df_dVSt = df_dVSt.reset_index()
        ax_Ntou[-1].plot(df_dVSt.TempC, df_dVSt.Total,  c = 'red', label = 'TOU High Price', linestyle='dashe
d' )

        df_NdVSt = df_tou1h_lPrice[df_tou1h_lPrice.GMT.str.contains('0' + str(i) + ':00:00')] #TOU demand vs
 temperature for low price periods
        df_NdVSt = df_NdVSt.groupby('TempC')['Total'].mean() / (df_NdVSt.shape[1] - 4)
        df_NdVSt = df_NdVSt.reset_index()
        ax_Ntou[-1].plot(df_NdVSt.TempC, df_NdVSt.Total, c = 'green', label = 'TOU Low Price', linestyle='das
hed' )

        df_NdVSt = df_tou1h_eventday_dPrice[df_tou1h_eventday_dPrice.GMT.str.contains('0' + str(i) + ':00:00'
)] #TOU demand vs temperature for default price periods
        df_NdVSt = df_NdVSt.groupby('TempC')['Total'].mean() / (df_NdVSt.shape[1] - 4)
        df_NdVSt = df_NdVSt.reset_index()
        ax_Ntou[-1].plot(df_NdVSt.TempC, df_NdVSt.Total, c = 'orange', label = 'TOU Default Price' )

        df_dVSt = df_tou1h_nf[df_tou1h_nf.GMT.str.contains('0' + str(i) + ':00:00')]
        df_dVSt = df_dVSt.groupby('TempC')['Total'].mean() / (df_dVSt.shape[1] - 4)
        df_dVSt = df_dVSt.reset_index()
        ax_Ntou[-1].plot(df_dVSt.TempC, df_dVSt.Total,  c = 'blue', label = 'TOU no-Event')# TOU demand vs te
mperature for non-Event

        ax_Ntou[-1].set_title('Time: 0' + str(i) + ':00')
        ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_Ntou[-1].set_ylabel('Average Consumption (kWh)')
        ax_Ntou[-1].legend()
    else:
        df_dVSt = df_tou1h_hPrice[df_tou1h_hPrice.GMT.str.contains(str(i) + ':00:00')] #TOU demand vs tempera
ture for high price periods
        df_dVSt = df_dVSt.groupby('TempC')['Total'].mean() / (df_dVSt.shape[1] - 4)
        df_dVSt = df_dVSt.reset_index()
        ax_Ntou[-1].plot(df_dVSt.TempC, df_dVSt.Total,  c = 'red', label = 'TOU High Price', linestyle='dashe
d')

        df_NdVSt = df_tou1h_lPrice[df_tou1h_lPrice.GMT.str.contains(str(i) + ':00:00')] #TOU demand vs temper
ature for low price periods
        df_NdVSt = df_NdVSt.groupby('TempC')['Total'].mean() / (df_NdVSt.shape[1] - 4)
        df_NdVSt = df_NdVSt.reset_index()
        ax_Ntou[-1].plot(df_NdVSt.TempC, df_NdVSt.Total, c = 'green', label = 'TOU Low Price', linestyle='das
hed' )

        df_NdVSt = df_tou1h_eventday_dPrice[df_tou1h_eventday_dPrice.GMT.str.contains(str(i) + ':00:00')] #TO
U demand vs temperature for default price periods
        df_NdVSt = df_NdVSt.groupby('TempC')['Total'].mean() / (df_NdVSt.shape[1] - 4)
        df_NdVSt = df_NdVSt.reset_index()
        ax_Ntou[-1].plot(df_NdVSt.TempC, df_NdVSt.Total, c = 'orange', label = 'TOU Default Price' )

        df_dVSt = df_tou1h_nf[df_tou1h_nf.GMT.str.contains(str(i) + ':00:00')]
        df_dVSt = df_dVSt.groupby('TempC')['Total'].mean() / (df_dVSt.shape[1] - 4)
        df_dVSt = df_dVSt.reset_index()
        ax_Ntou[-1].plot(df_dVSt.TempC, df_dVSt.Total,  c = 'blue', label = 'TOU no-Event' )# TOU demand vs t
emperature for non-Event

        ax_Ntou[-1].set_title('Time: ' + str(i) + ':00')
        ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_Ntou[-1].set_ylabel('Average Consumption (kWh)')
        ax_Ntou[-1].legend()
plt.tight_layout()
```

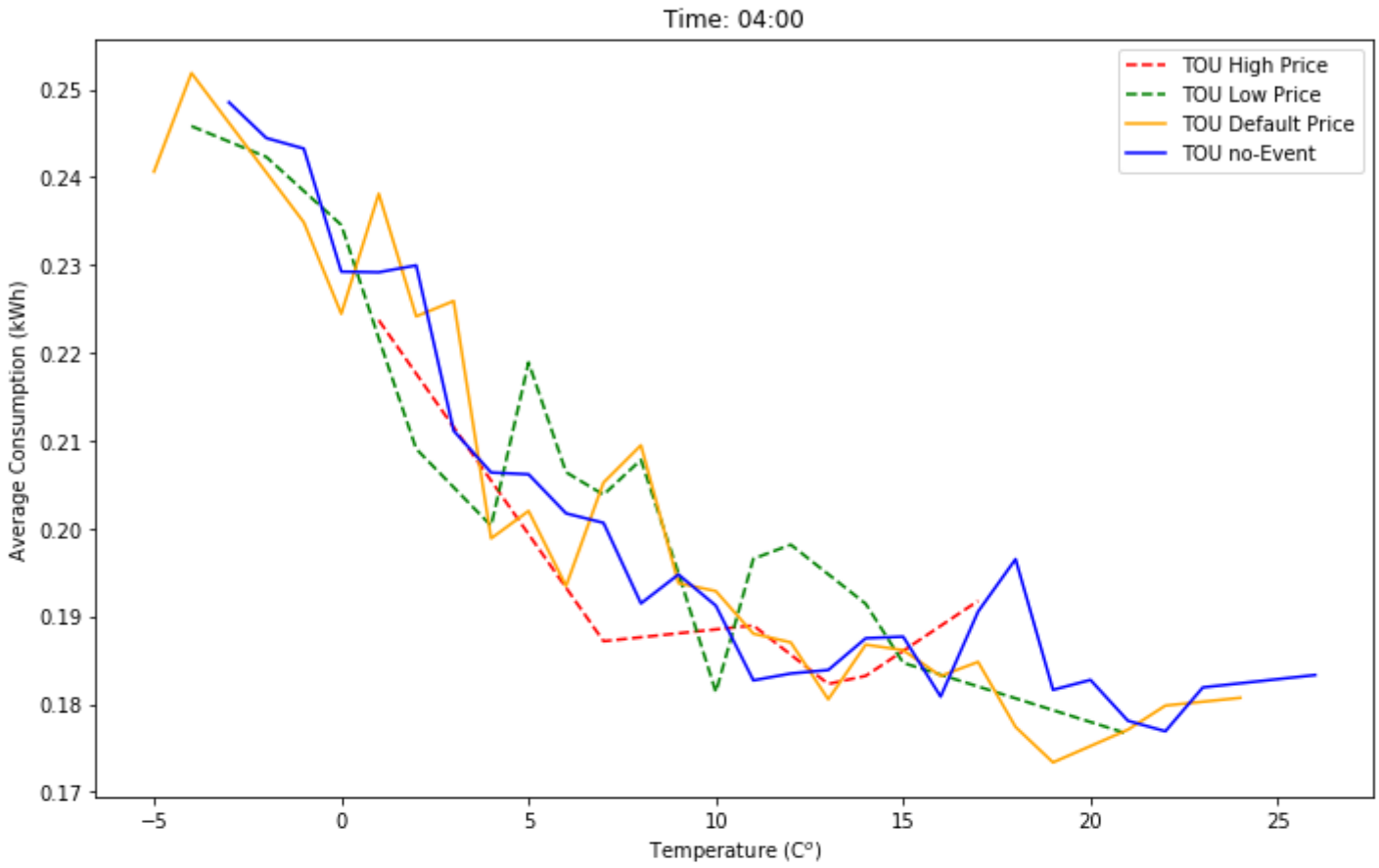

Time: 00:00
TOU High Price
TOU Low Price
TOU Default Price
TOU no-Event
Average Consumption (kWh)
Temperature (C°)

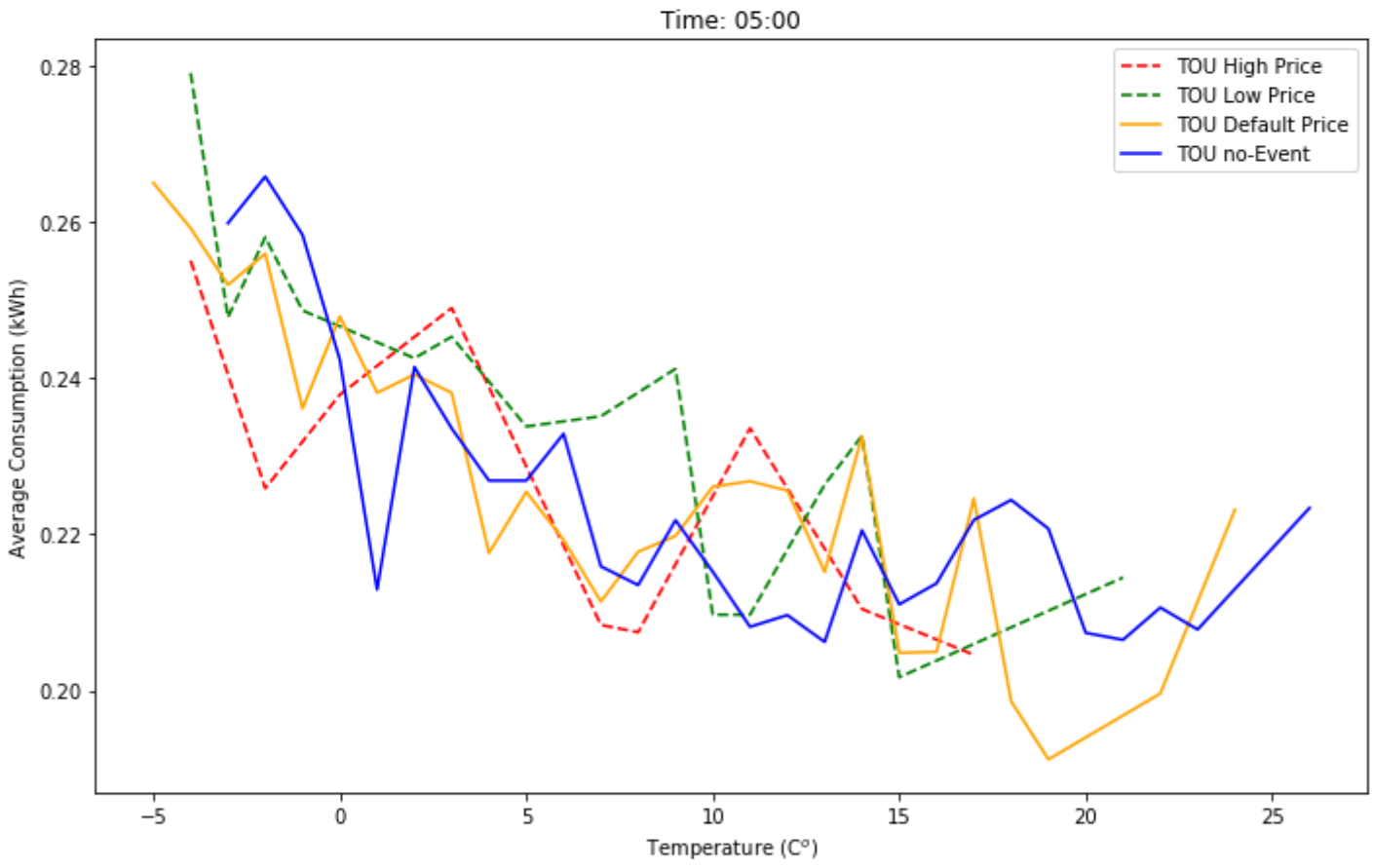

Time: 01:00
TOU High Price
TOU Low Price
TOU Default Price
TOU no-Event
Average Consumption (kWh)
Temperature (C°)

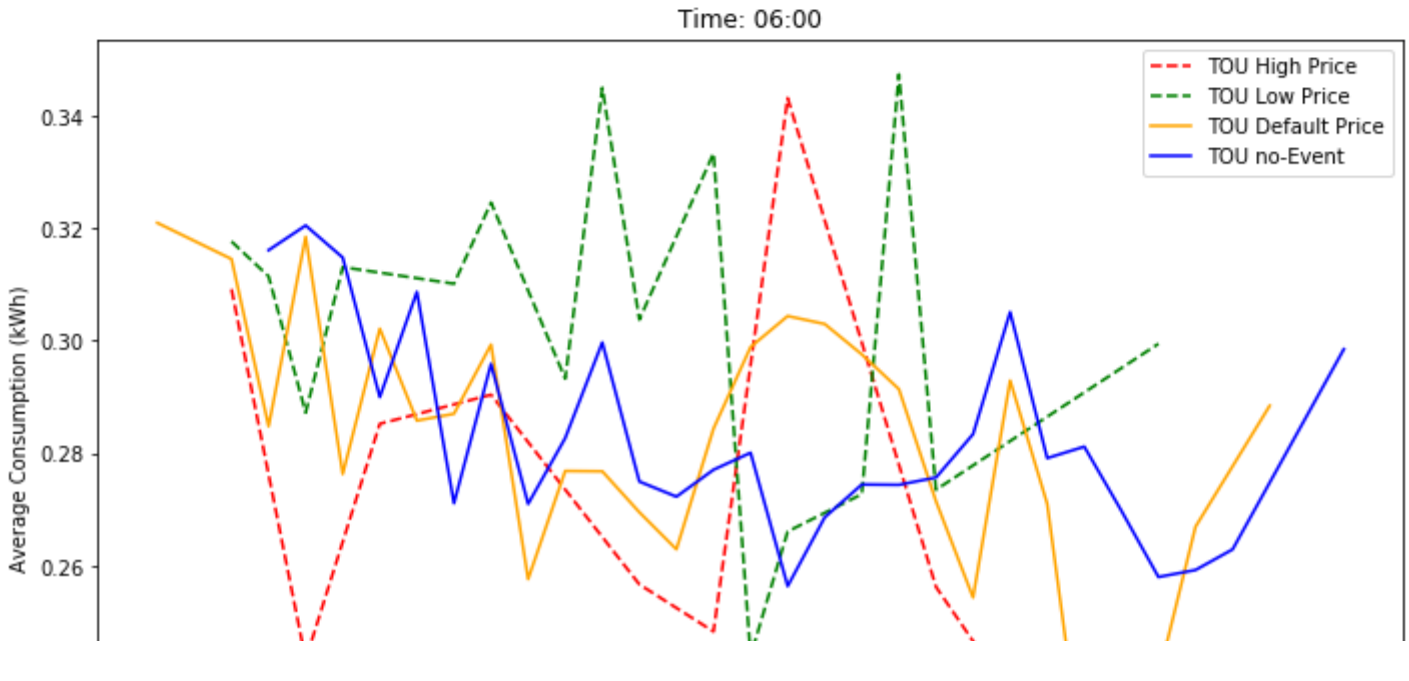

Time: 02:00
TOU High Price
TOU Low Price
TOU Default Price
TOU no-Event
Average Consumption (kWh)
Temperature (C°)

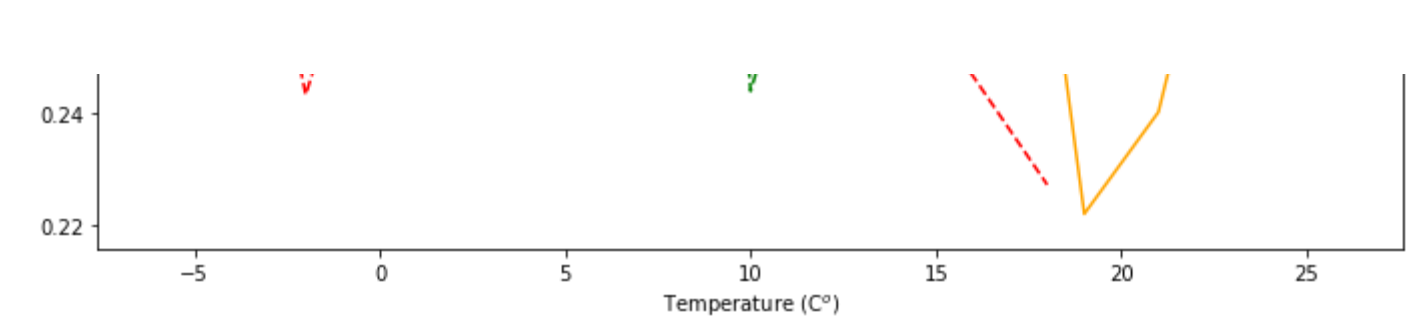
Time: 03:00
TOU High Price
TOU Low Price
TOU Default Price
TOU no-Event

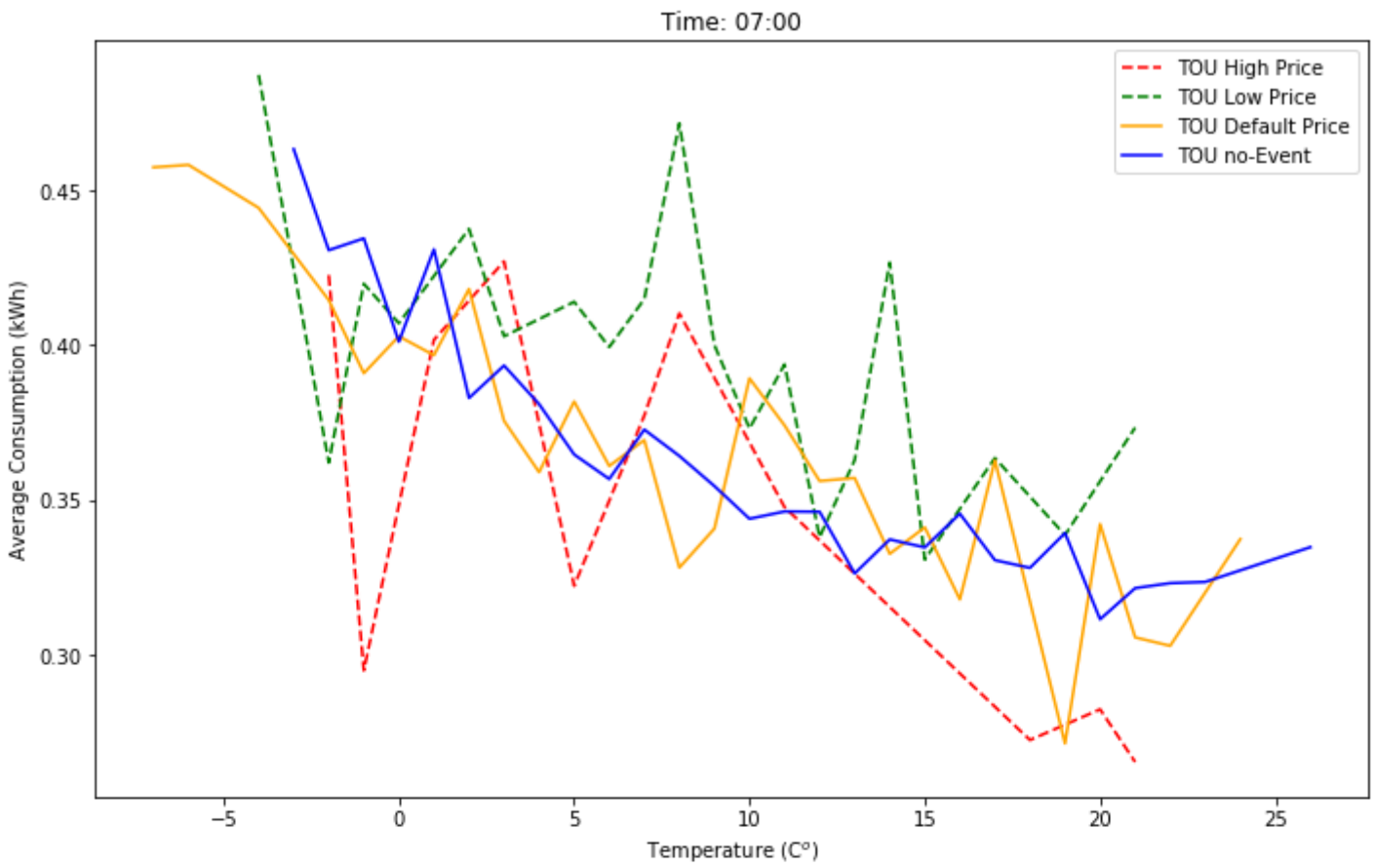

Average Consumption (kW
0.22
0.20
0.18
−5
0
5
10
15
20
25
Temperature (C°)

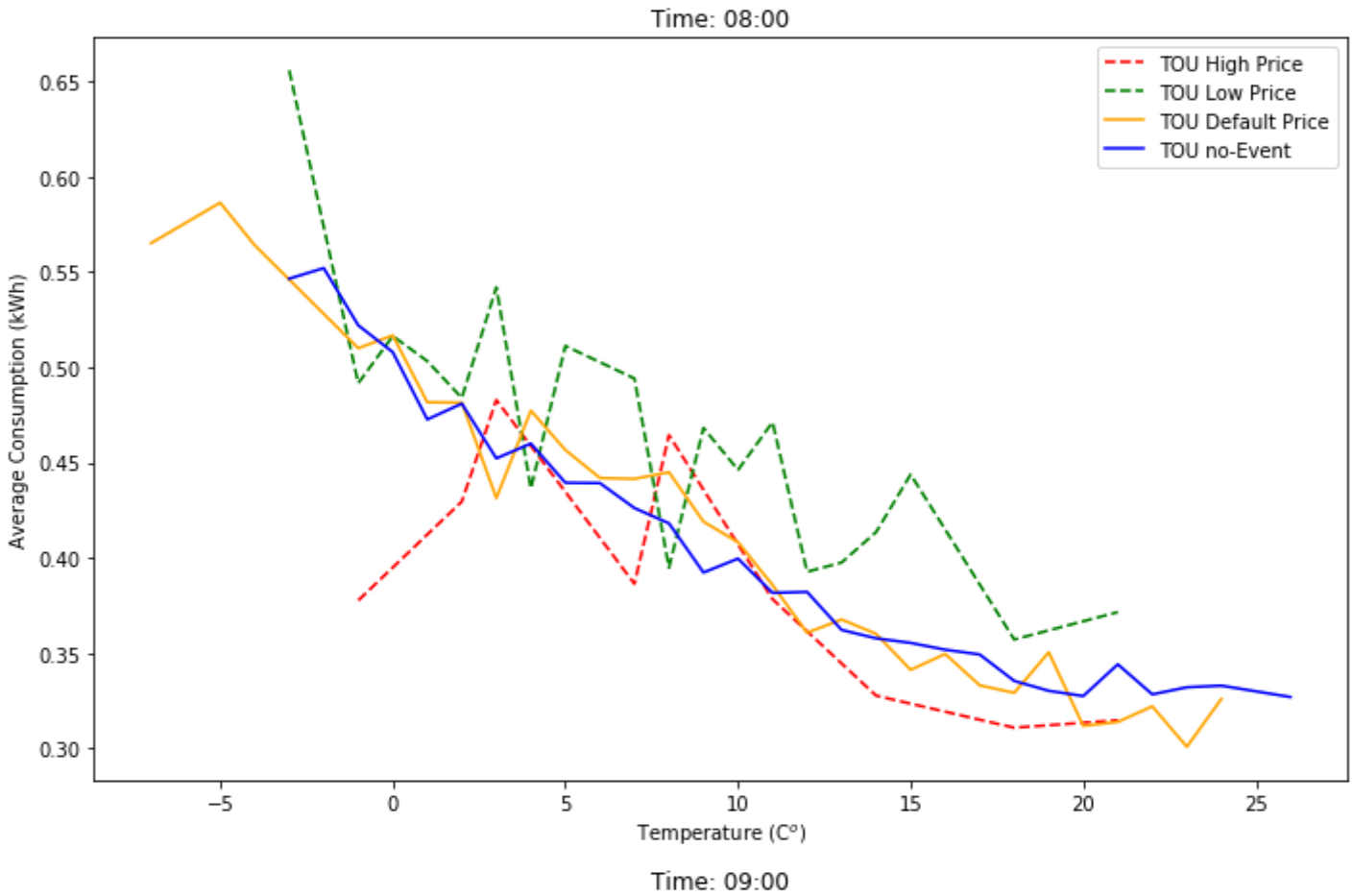

Time: 04:00
TOU High Price
TOU Low Price
TOU Default Price
TOU no-Event
ion (kWh)
Average Consumption (kWh)
0.25
0.24
0.23
0.22
0.21
0.20
0.19
0.18
0.17
−5
0
5
10
15
20
25
Temperature (C°)

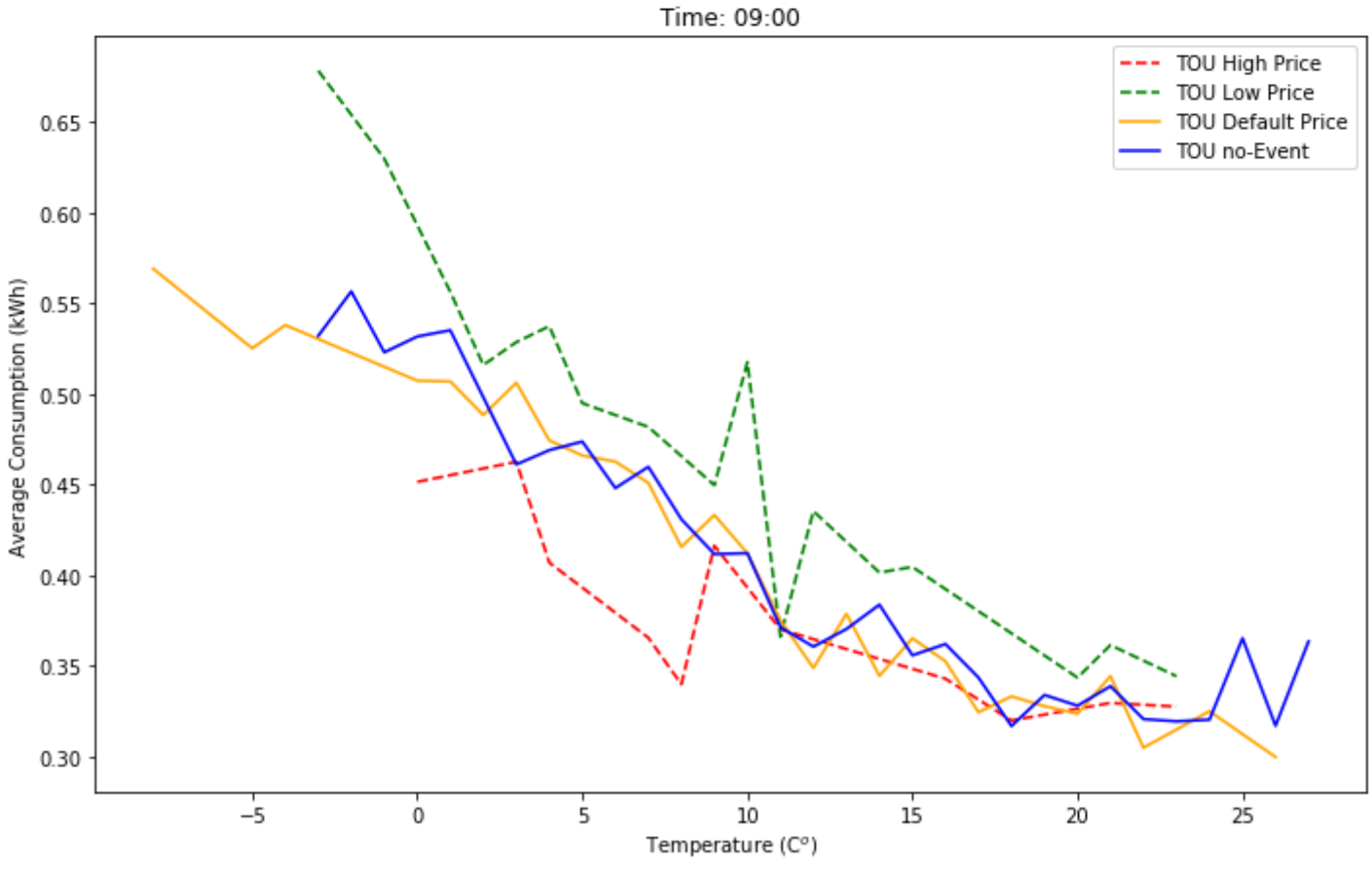

Time: 05:00
TOU High Price
TOU Low Price
TOU Default Price
TOU no-Event
Average Consumption (kWh)
0.28
0.26
0.24
0.22
0.20
−5
0
5
10
15
20
25
Temperature (C°)

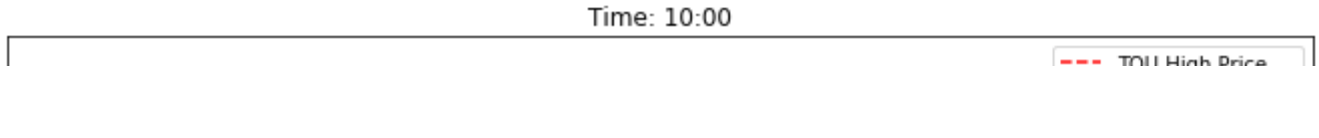

Time: 06:00
TOU High Price
TOU Low Price
TOU Default Price
TOU no-Event
Average Consumption (kWh)
0.34
0.32
0.30
0.28
0.26

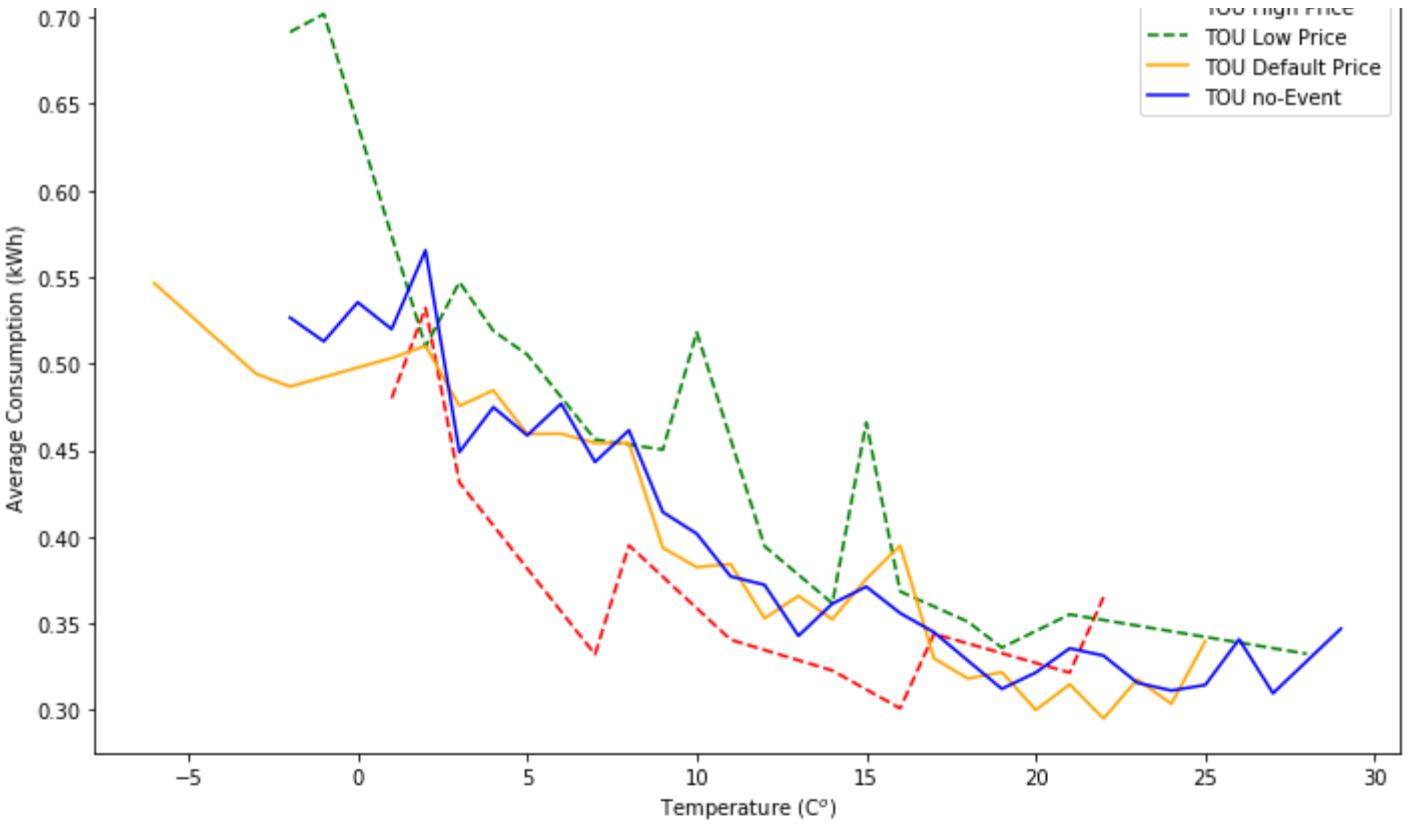

0.24
0.22
−5
0
5
10
15
20
25
Temperature (C°)


Time: 07:00
TOU High Price
TOU Low Price
TOU Default Price
TOU no-Event
Average Consumption (kWh)
0.45
0.40
0.35
0.30
−5
0
5
10
15
20
25
Temperature (C°)

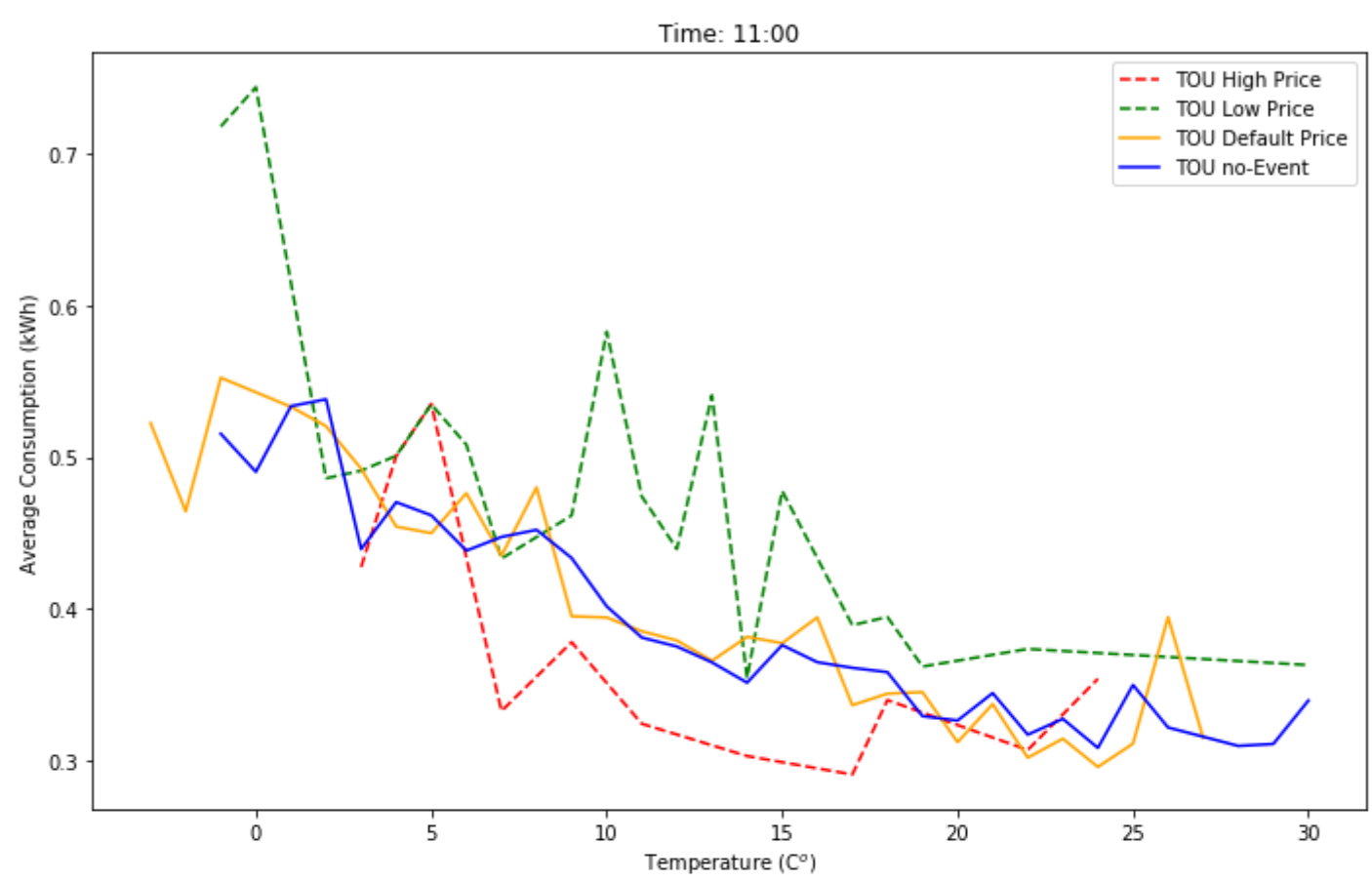

Time: 08:00
TOU High Price
TOU Low Price
TOU Default Price
TOU no-Event
Average Consumption (kWh)
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
−5
0
5
10
15
20
25
Temperature (C°)


Time: 09:00
TOU High Price
TOU Low Price
TOU Default Price
TOU no-Event
Average Consumption (kWh)
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
−5
0
5
10
15
20
25
Temperature (C°)

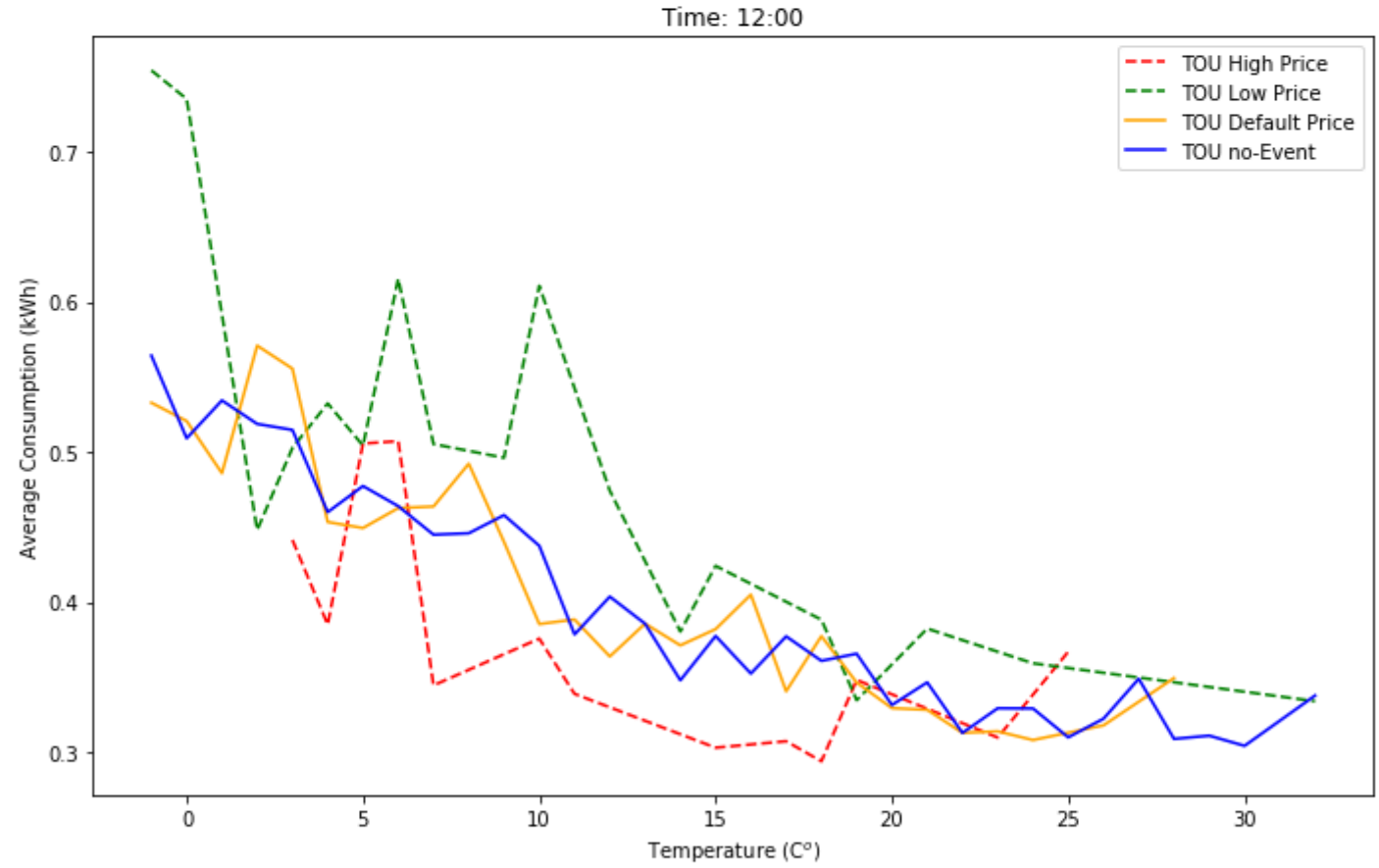

Time: 10:00



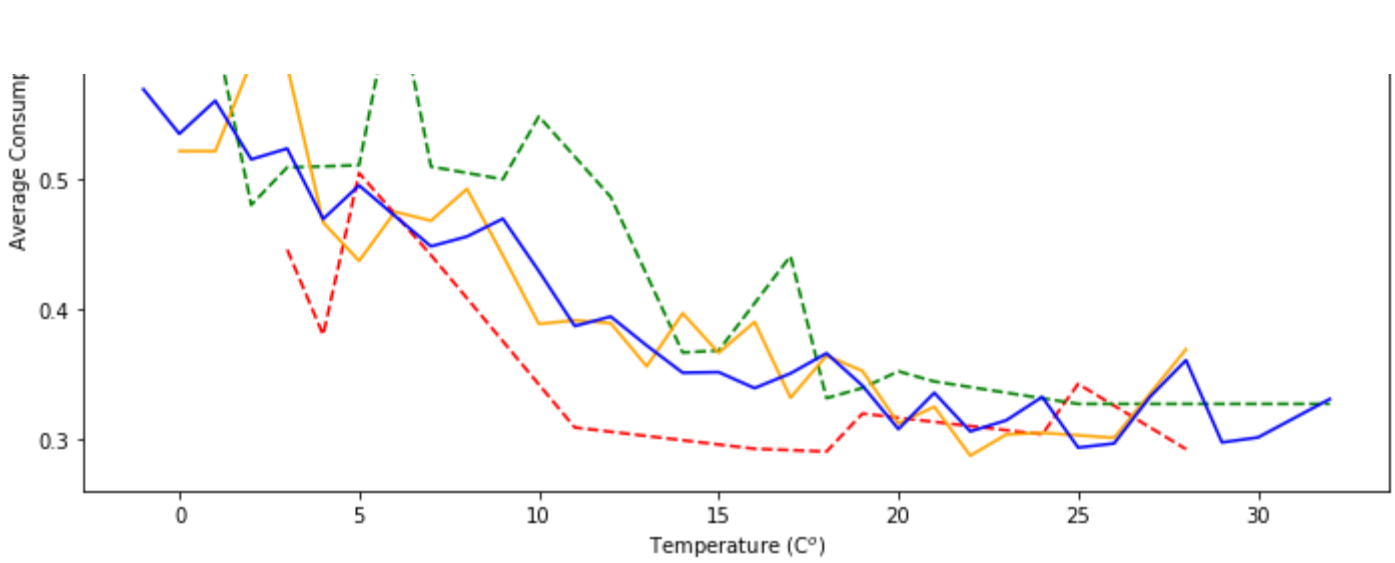
Time: 11:00

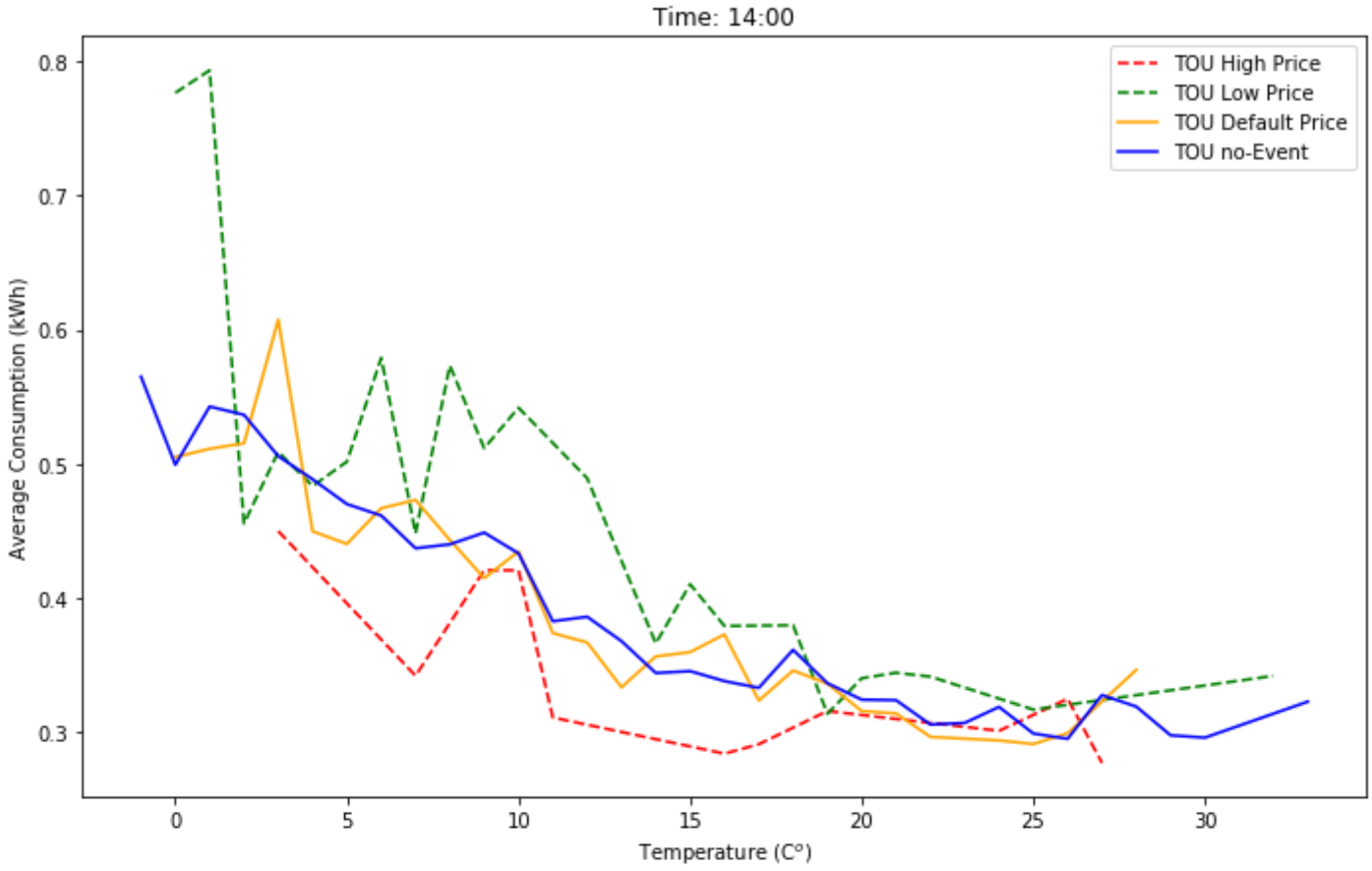


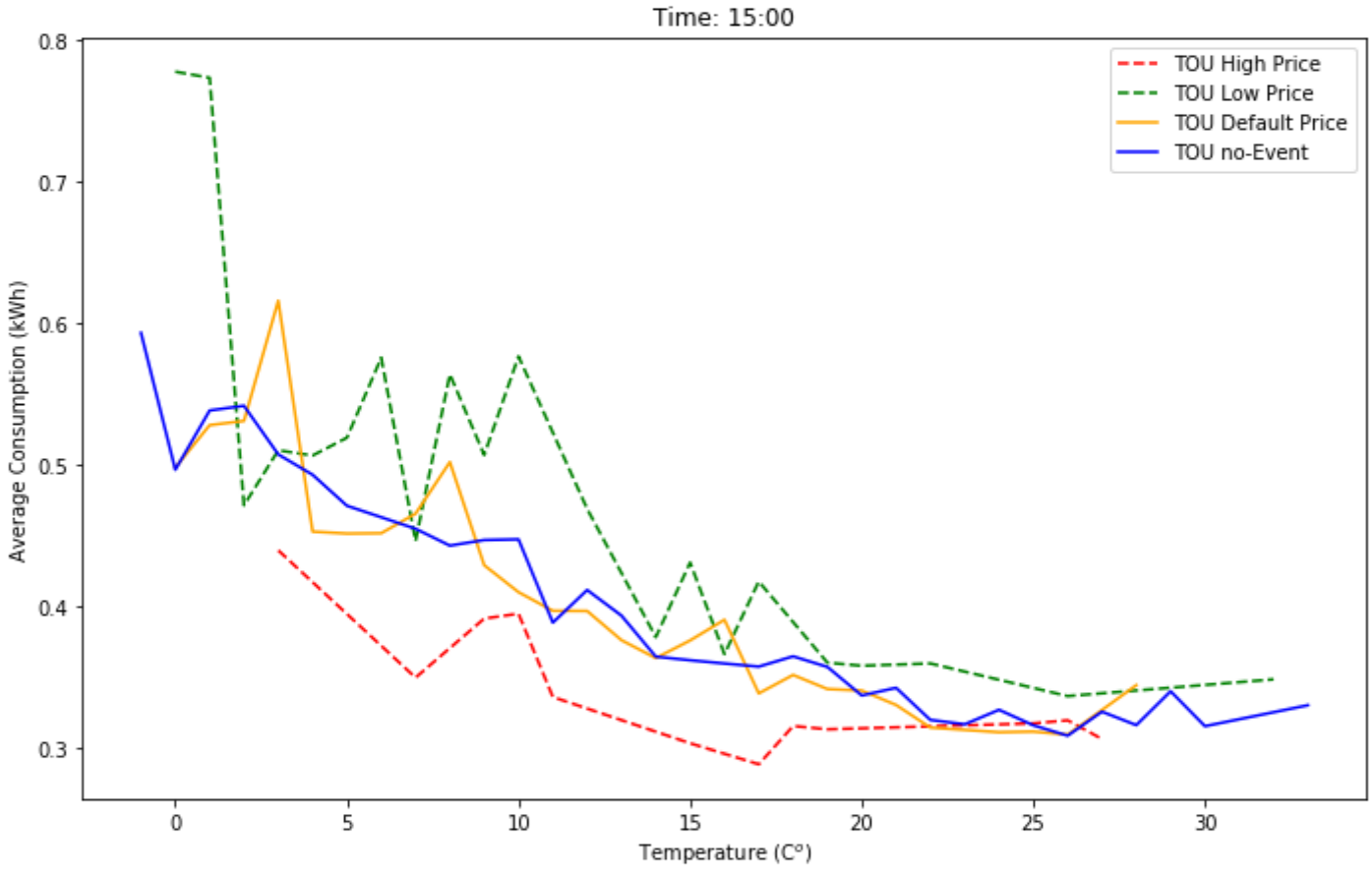
Time: 12:00

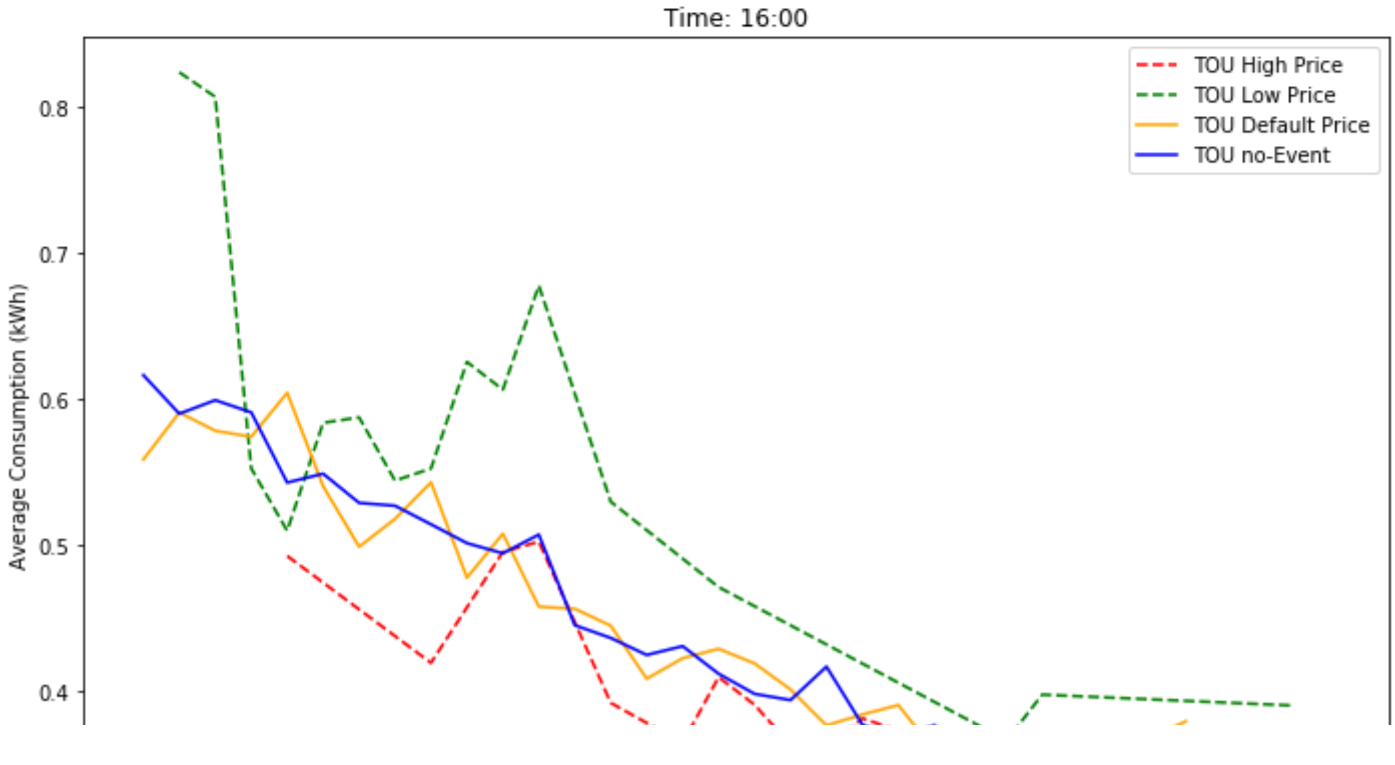


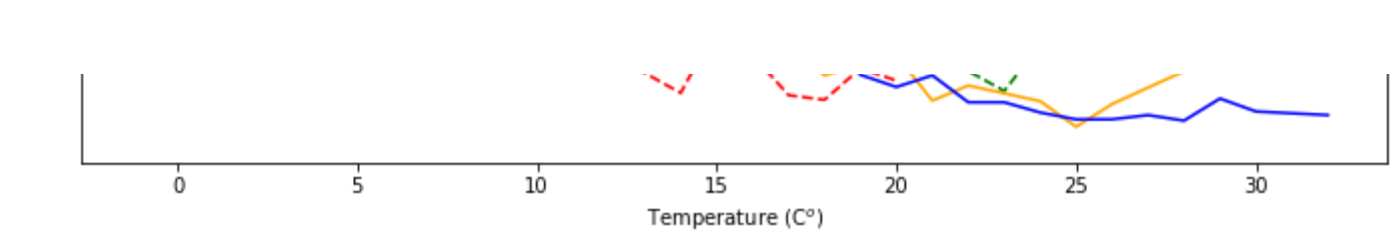
Time: 13:00

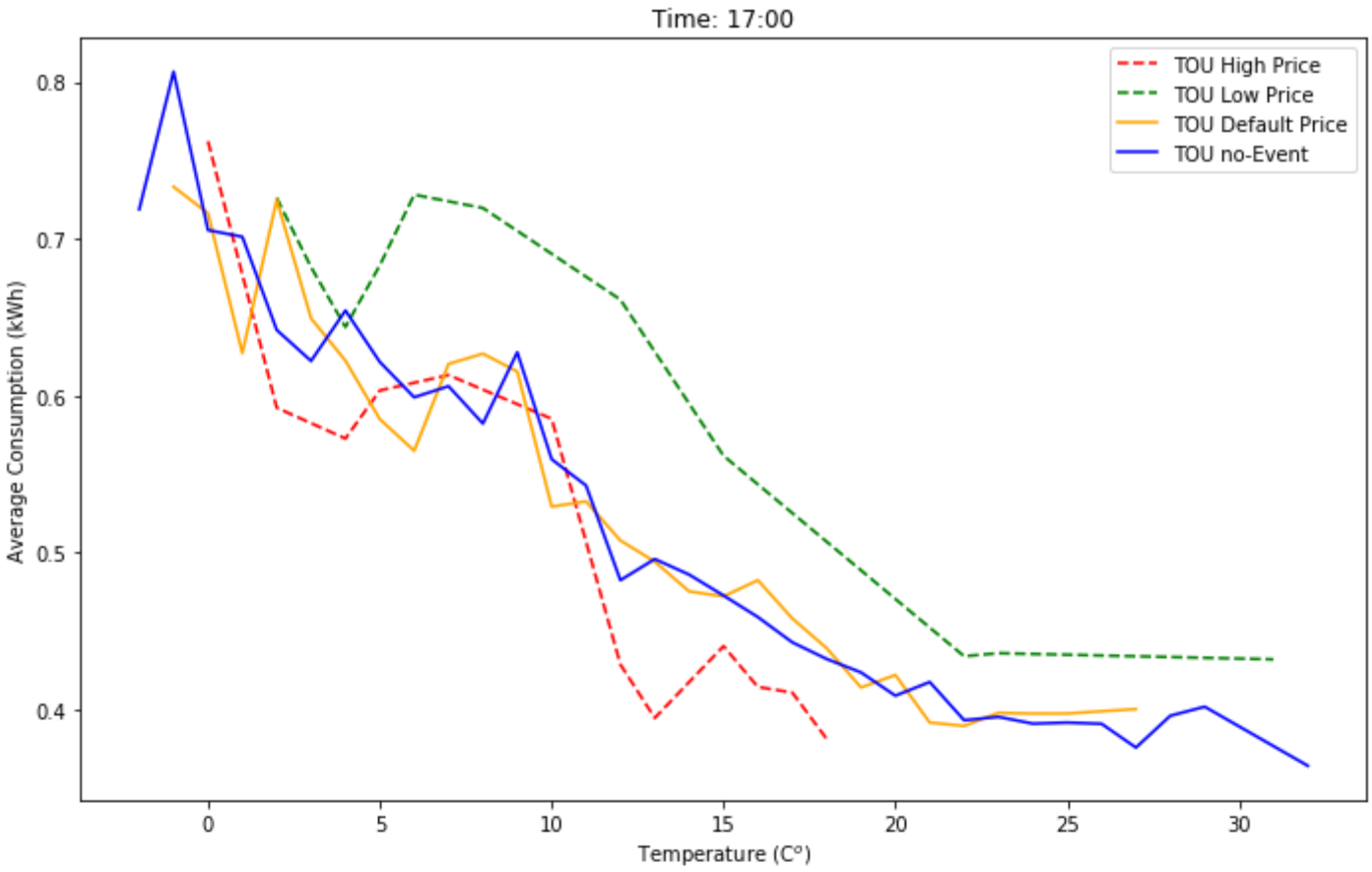

Average Consump
0.5
0.4
0.3
0
5
10
15
20
25
30
Temperature (C°)

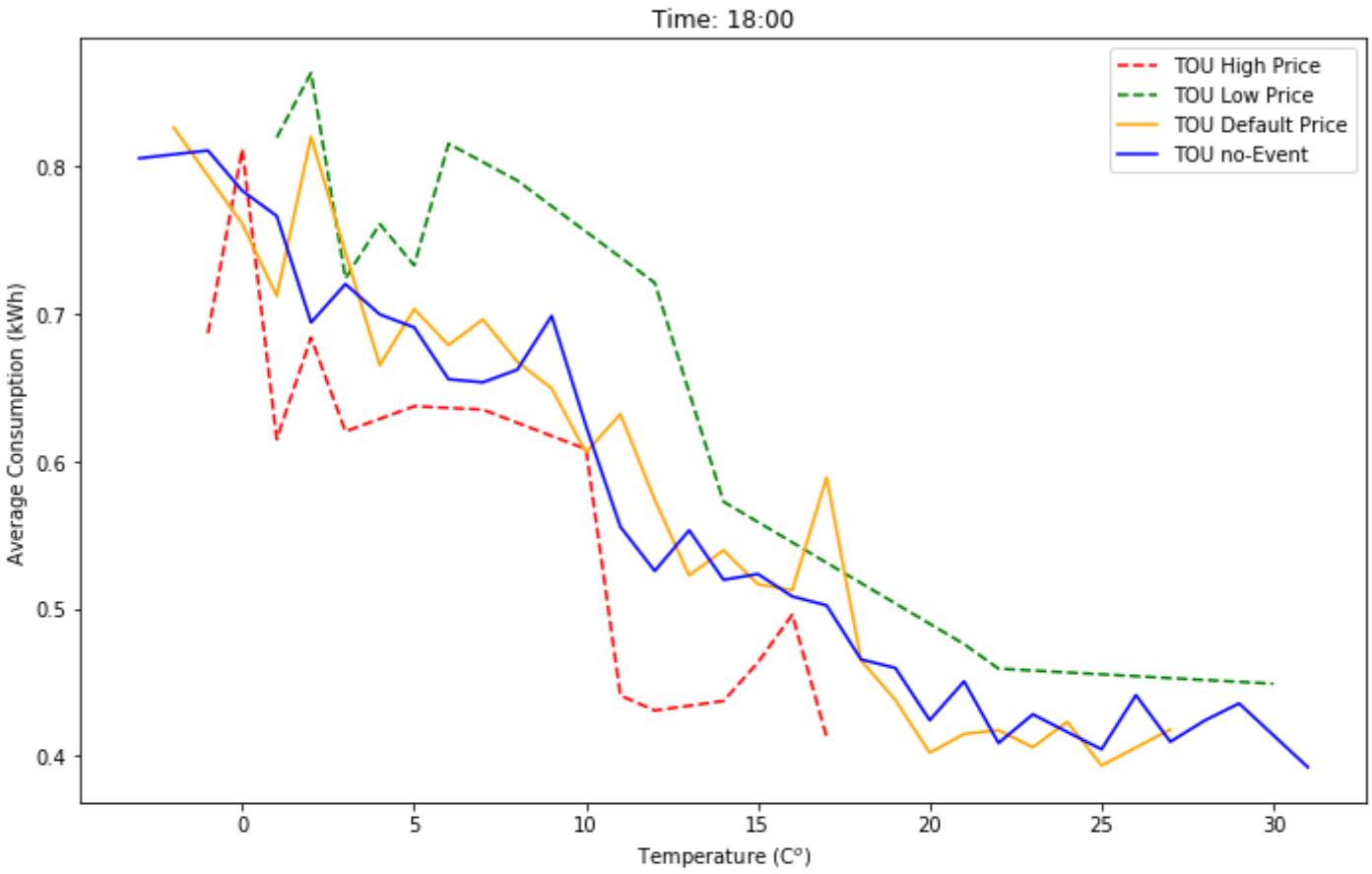

Time: 14:00
TOU High Price
TOU Low Price
TOU Default Price
TOU no-Event
Average Consumption (kWh)
0.8
0.7
0.6
0.5
0.4
0.3
0
5
10
15
20
25
30
Temperature (C°)

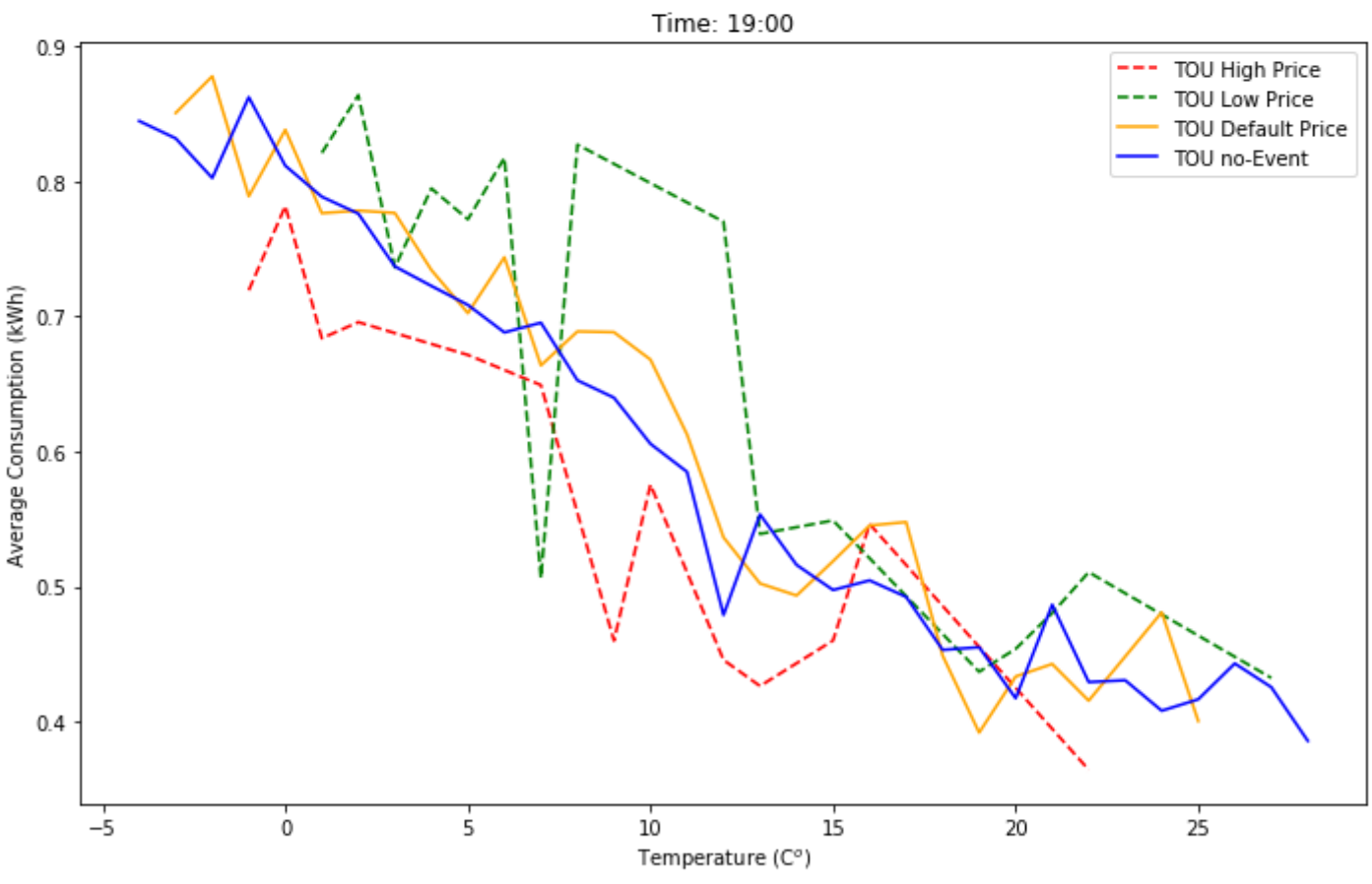

Time: 15:00
TOU High Price
TOU Low Price
TOU Default Price
TOU no-Event
Average Consumption (kWh)
0.8
0.7
0.6
0.5
0.4
0.3
0
5
10
15
20
25
30
Temperature (C°)


Time: 16:00
TOU High Price
TOU Low Price
TOU Default Price
TOU no-Event
Average Consumption (kWh)
0.8
0.7
0.6
0.5
0.4

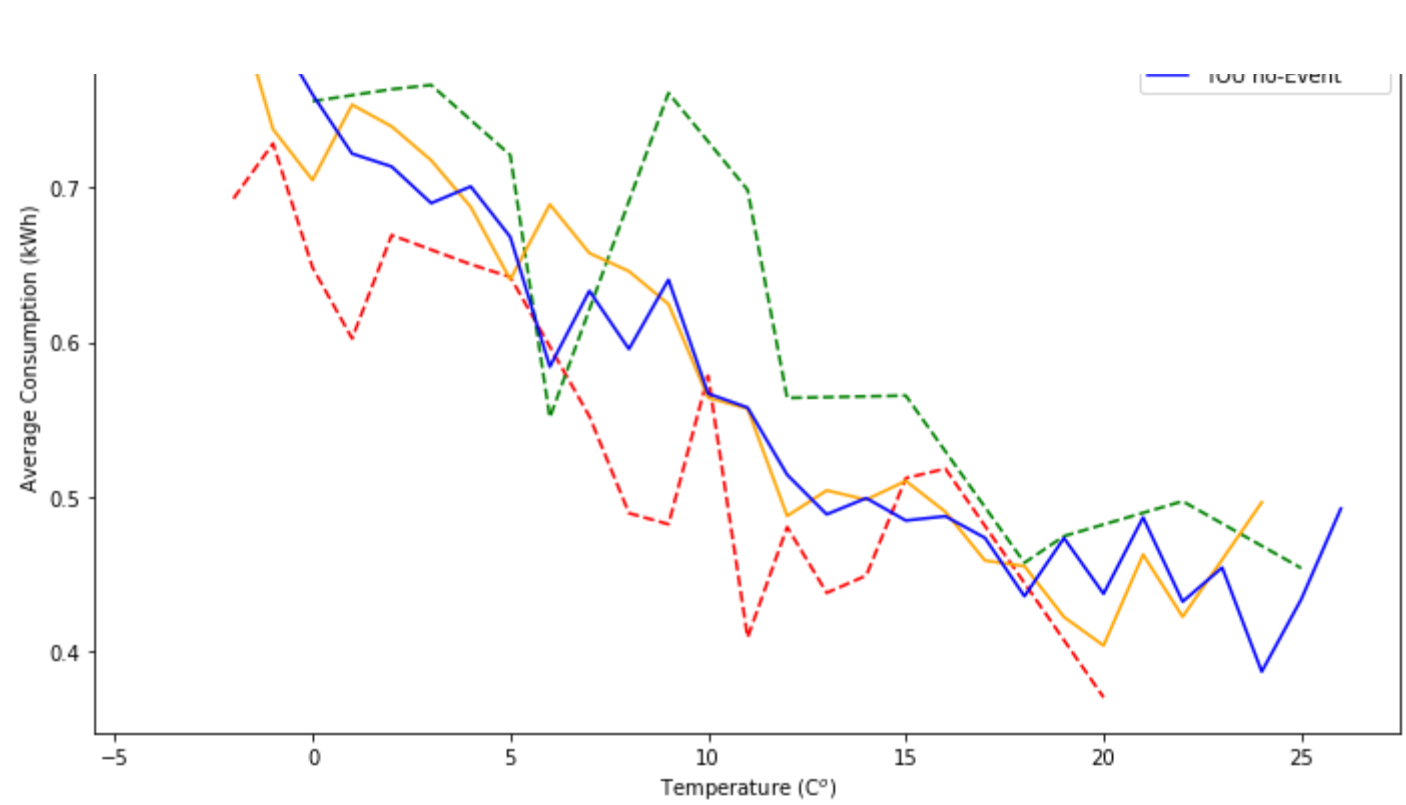

0
5
10
15
20
25
30
Temperature (C°)

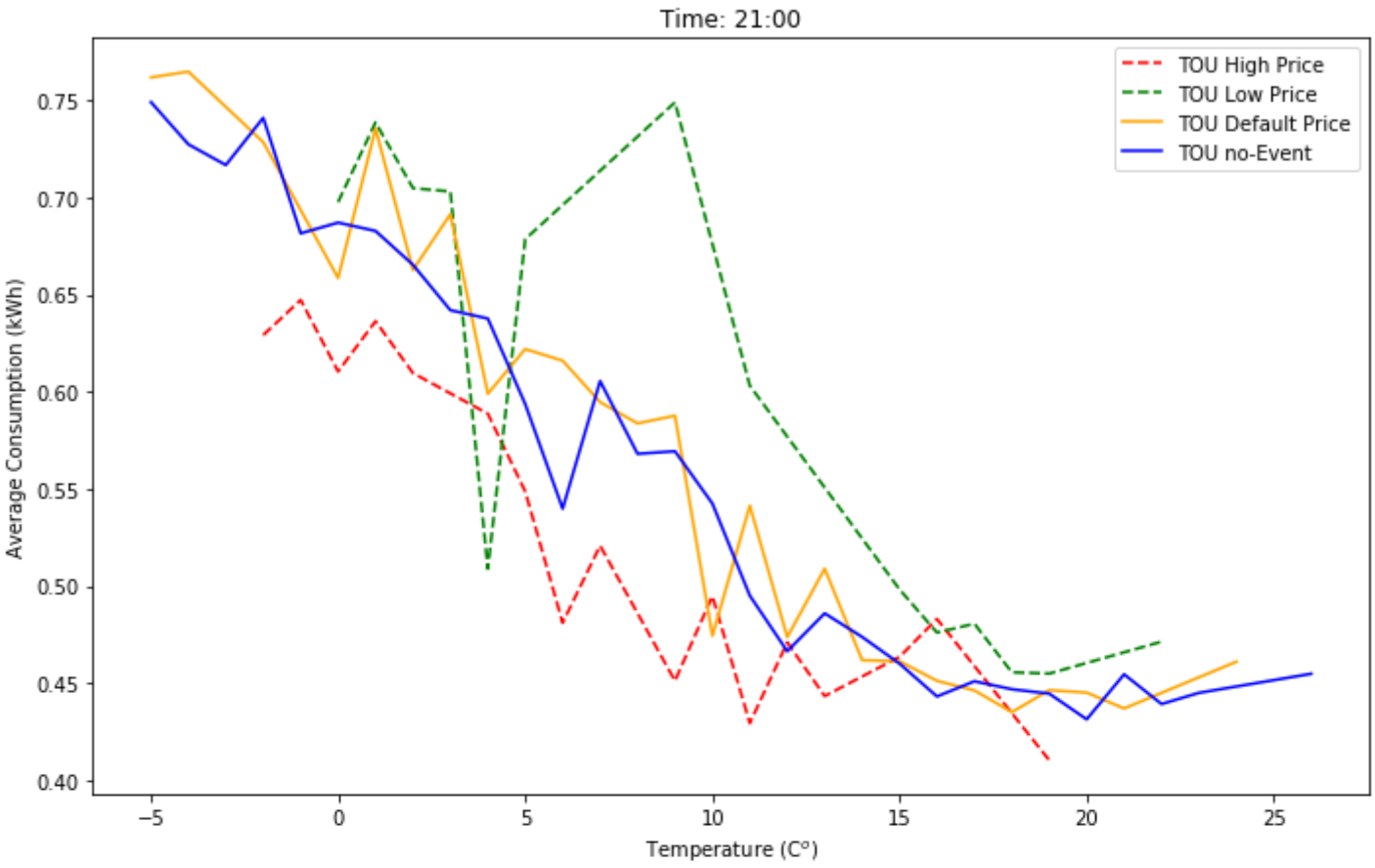

Time: 17:00
TOU High Price
TOU Low Price
TOU Default Price
TOU no-Event
Average Consumption (kWh)
0.8
0.7
0.6
0.5
0.4
0
5
10
15
20
25
30
Temperature (C°)

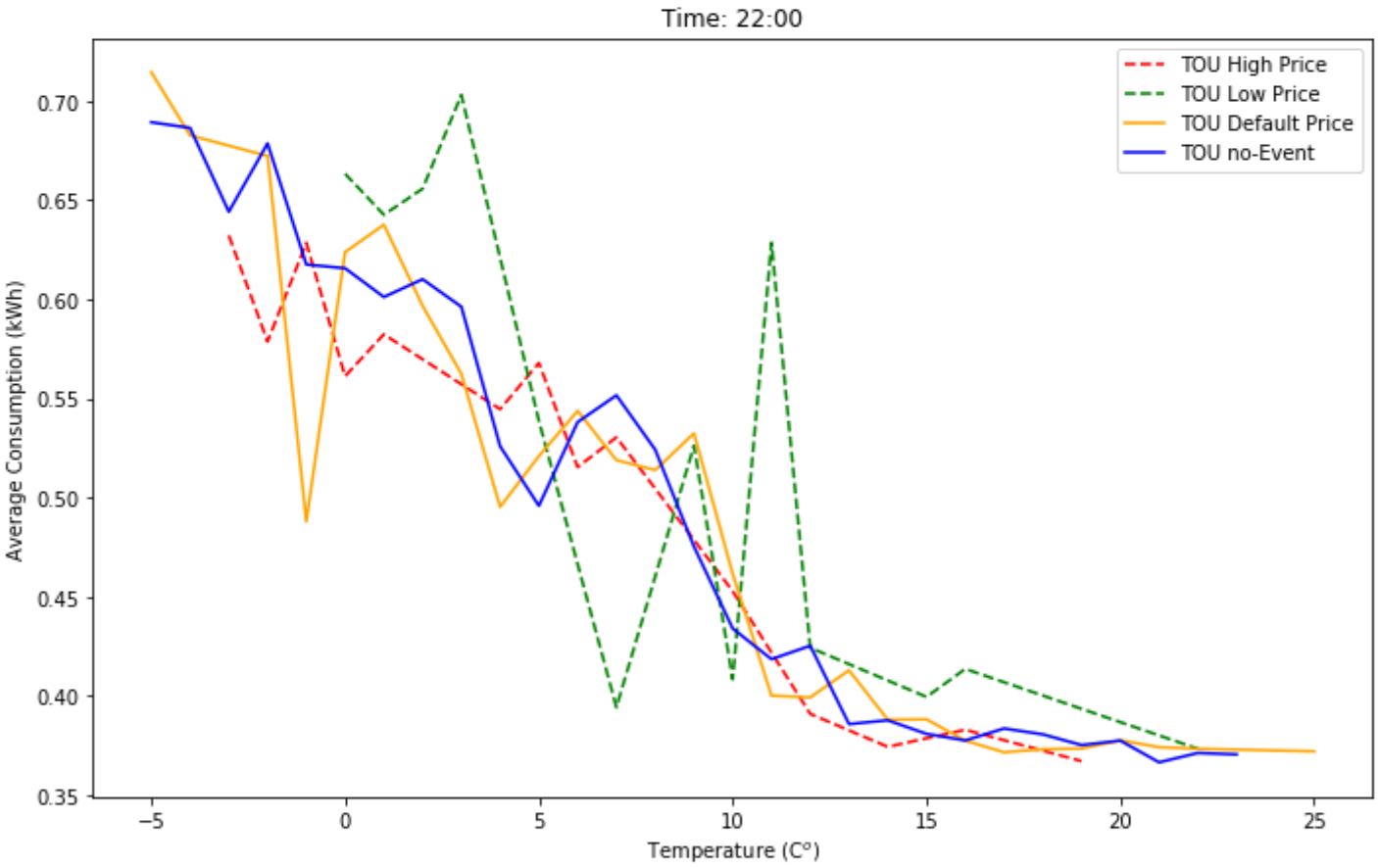

Time: 18:00
TOU High Price
TOU Low Price
TOU Default Price
TOU no-Event
Average Consumption (kWh)
0.8
0.7
0.6
0.5
0.4
0
5
10
15
20
25
30
Temperature (C°)

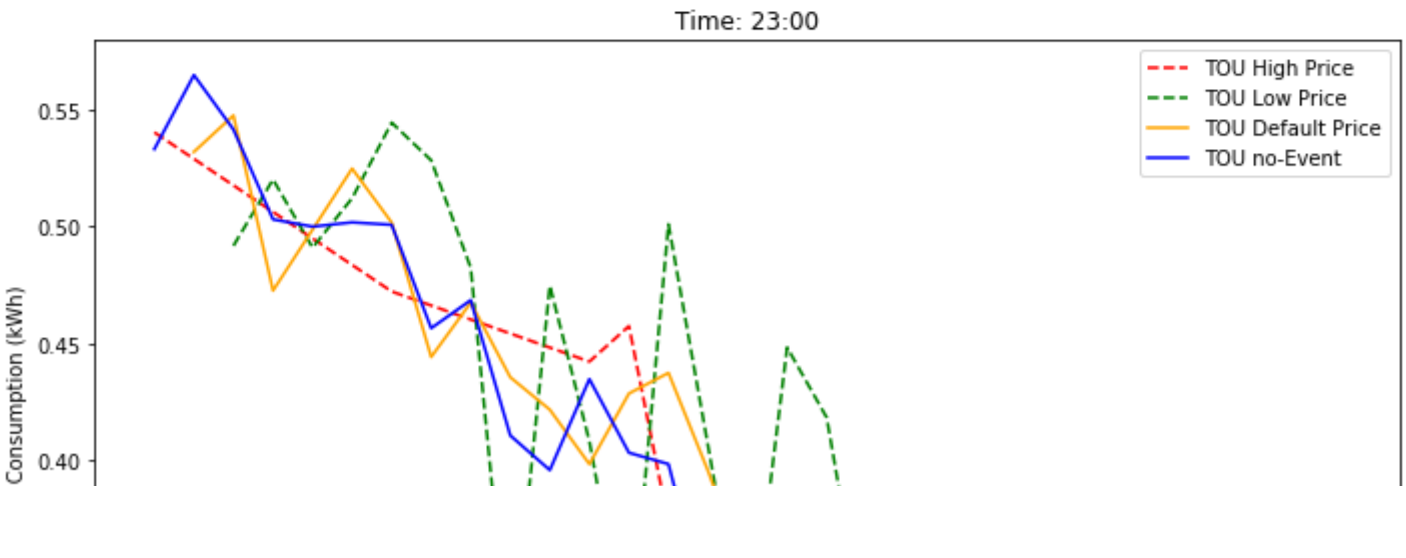

Time: 19:00
TOU High Price
TOU Low Price
TOU Default Price
TOU no-Event
Average Consumption (kWh)
0.9
0.8
0.7
0.6
0.5
0.4
−5
0
5
10
15
20
25
Temperature (C°)

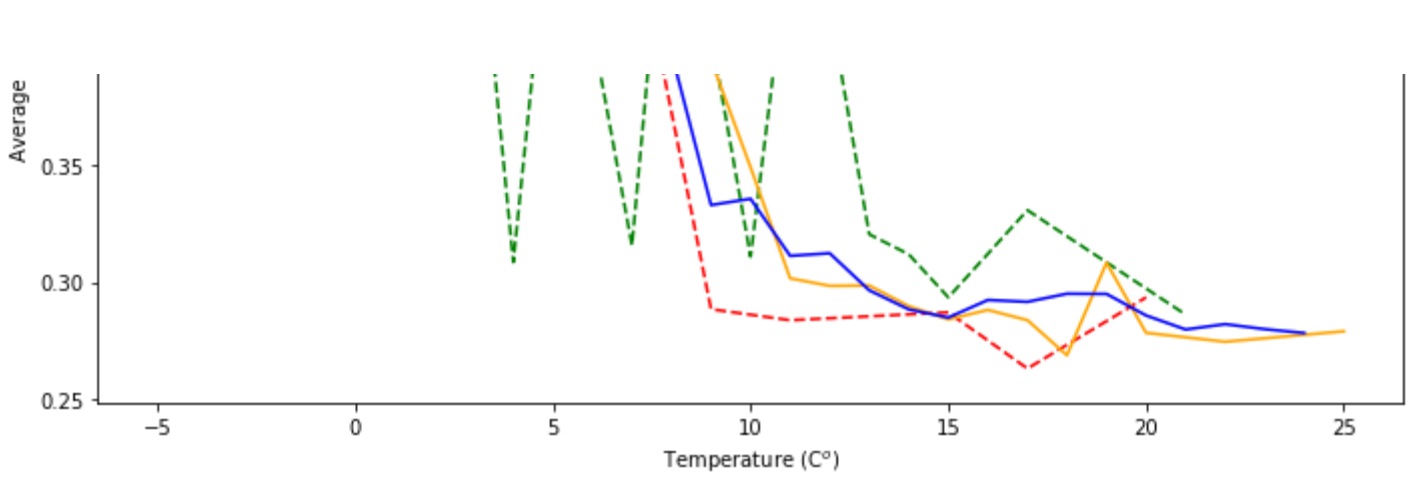

Time: 20:00
TOU High Price
TOU Low Price
TOU Default Price
0.8


TOU no-Event
Average Consumption (kWh)
0.7
0.6
0.5
0.4
−5
0
5
10
15
20
25
Temperature (C°)


Time: 21:00
TOU High Price
TOU Low Price
TOU Default Price
TOU no-Event
Average Consumption (kWh)
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
−5
0
5
10
15
20
25
Temperature (C°)


Time: 22:00
TOU High Price
TOU Low Price
TOU Default Price
TOU no-Event
Average Consumption (kWh)
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
−5
0
5
10
15
20
25
Temperature (C°)


Time: 23:00
TOU High Price
TOU Low Price
TOU Default Price
TOU no-Event
Consumption (kWh)
0.55
0.50
0.45
0.40



The above figures tells us, the high price (red) usually appears in daytime or evening, and seldomly appears in a midnight (can add figures to visualize this) since red curves have less turning points, thus possible data points are less during midnight for the high pricing periods.

Also, during midnight, high pricing seems not to affect the average consumption too much (probably because people go to sleep, not too much consumption needed thus small flexibility on energy consumption at this time)

But from 9am to 11pm, the red curves are in general below blue curve, meaning that during the daytime(late morning) and evening, people's behavior are affected by high pricing, i.e., people tends to decrease their consumption.

Interestingly, for low pricing, it seems to work over all day night and more effective than high pricing in terms of affecting user's consumption. People tend to increase their consumption in general during low pricing. During daytime and evening it works best, for midnight it still has effect and more significant than high pricing.

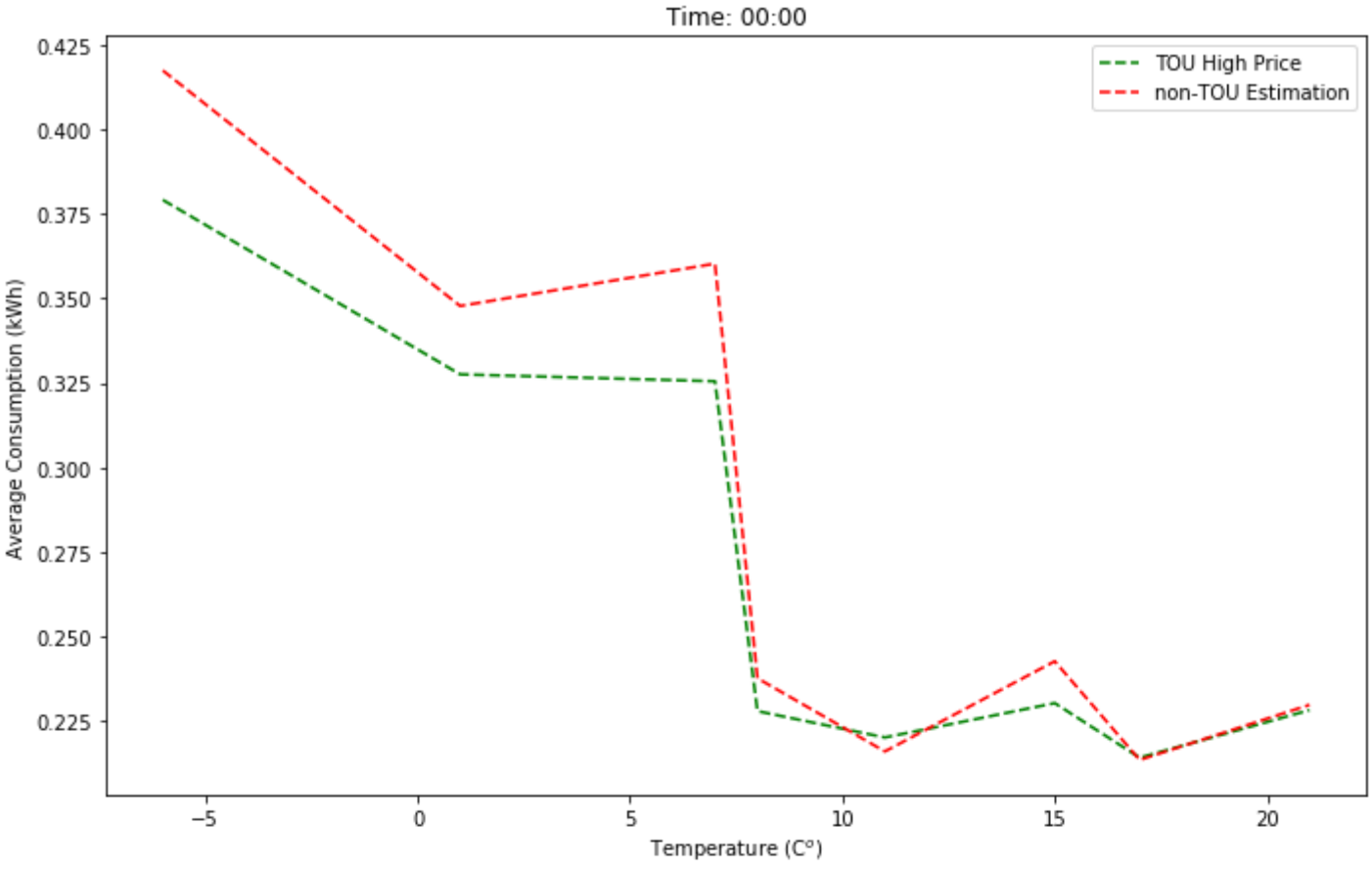
The third thing needs to be mentioned is that around 6am, things become unclear, so 6am seems to be involved in more randomness of behavior. We could look deeper on that.

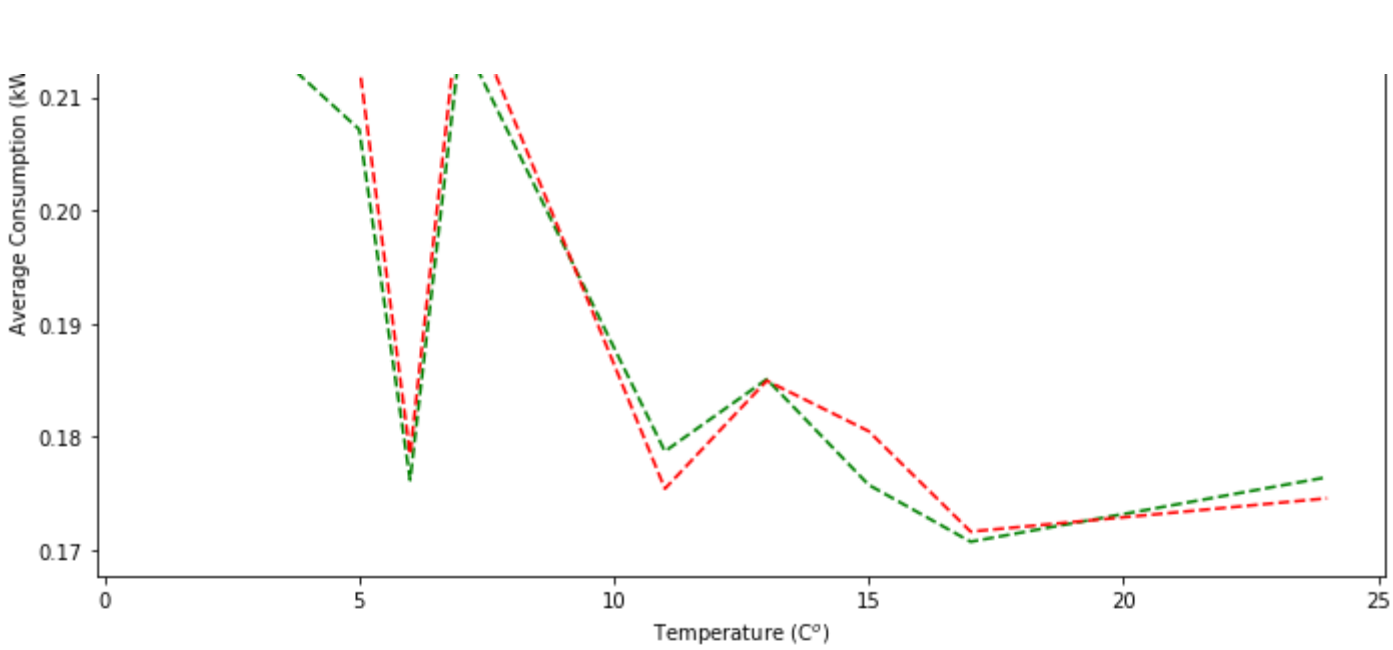
Using non-TOU data with the constant shift to compare with the TOU data at the same flexible (event) time periods.

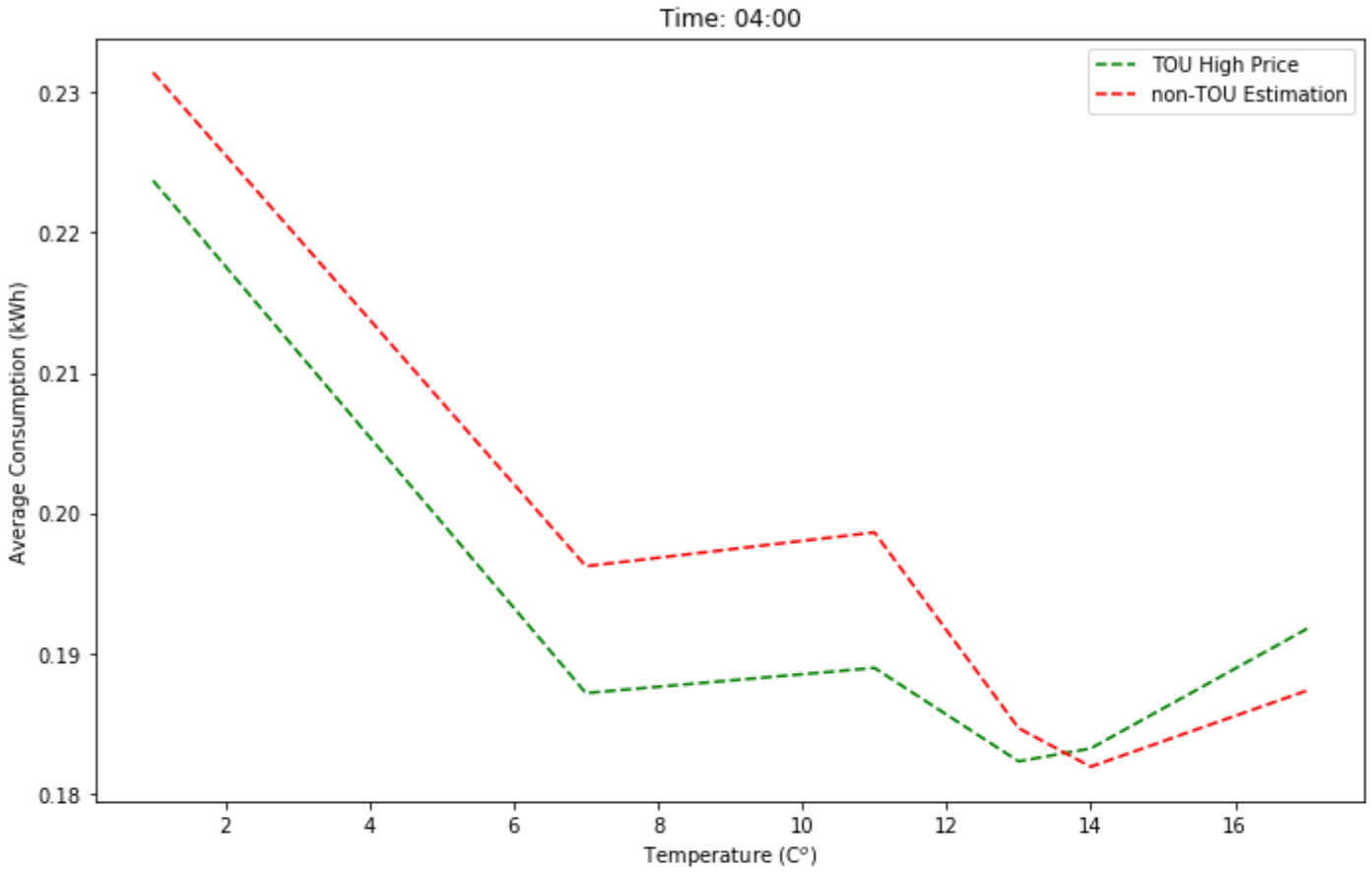
Note that this cannot be done with TOU data, since at the same time and day of the event time, TOU group all facing the same price, it's impossible to compare with them at this time with a lower price, but we can compare with a counterpart with removing the constant difference.

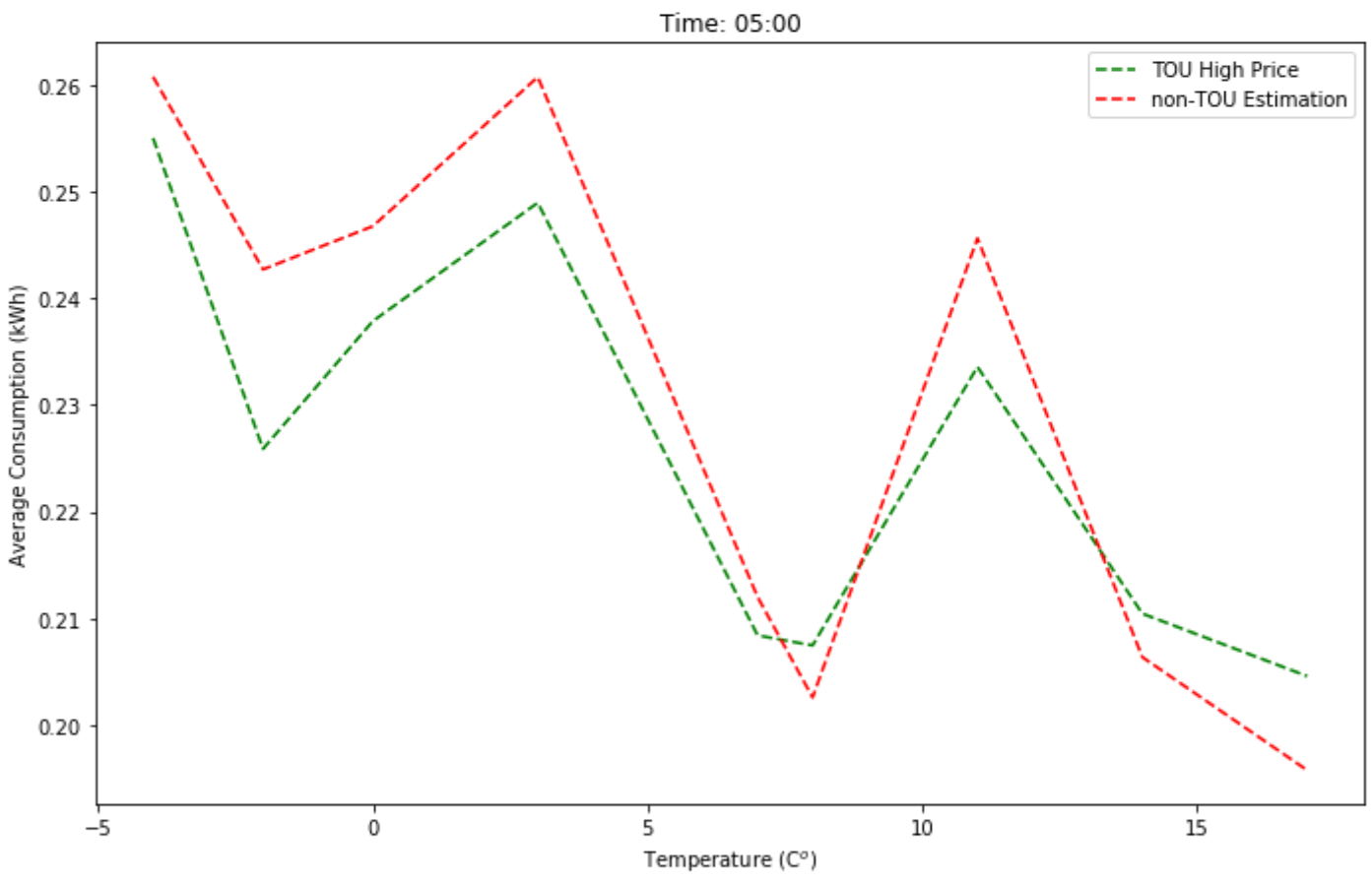
In [87]:
```
# Calculate the average shift between TOU and non-TOU in non-event time, which reflects their intrinsic diffe
rence.
# non-TOU vs TOU in terms of average (over both users and the time) hourly energy consumption v.s. temperatur
e in non-Event days
const_shift = [] # store constant shift between non-TOU and TOU, note: in this case it's non-TOU minus TOU
for i in range(24):
    if i <= 9:
        # calculate the average consumption grouped by different temperatures
        df_NdVSt = df_Ntou1h_nf[df_Ntou1h_nf.GMT.str.contains('0' + str(i) + ':00:00')] #non-tou demand vs te
mperature for non-Event
        df_NdVSt = df_NdVSt.groupby('TempC')['Total'].mean() / (df_NdVSt.shape[1] - 4)
        df_NdVSt = df_NdVSt.reset_index()

        df_dVSt = df_tou1h_nf[df_tou1h_nf.GMT.str.contains('0' + str(i) + ':00:00')]
        df_dVSt = df_dVSt.groupby('TempC')['Total'].mean() / (df_dVSt.shape[1] - 4)
        df_dVSt = df_dVSt.reset_index()

        const_shift.append((df_NdVSt.Total - df_dVSt.Total).mean())
    else:
        df_NdVSt = df_Ntou1h_nf[df_Ntou1h_nf.GMT.str.contains(str(i) + ':00:00')]
        df_NdVSt = df_NdVSt.groupby('TempC')['Total'].mean() / (df_NdVSt.shape[1] - 4)
        df_NdVSt = df_NdVSt.reset_index()

        df_dVSt = df_tou1h_nf[df_tou1h_nf.GMT.str.contains(str(i) + ':00:00')]
        df_dVSt = df_dVSt.groupby('TempC')['Total'].mean() / (df_dVSt.shape[1] - 4)
        df_dVSt = df_dVSt.reset_index()

        const_shift.append((df_NdVSt.Total - df_dVSt.Total).mean())
```

In [102]:
```
# TOU high price vs estimation of TOU group facing default price using non-TOU data
# hourly energy consumption v.s. temperature in event days
fig_all = plt.figure(figsize = (10,150))
ax_Ntou = [] # store subplot objects
ax_tou = [] # store subplot objects
for i in range(24):
    ax_Ntou.append(fig_all.add_subplot(24, 1, i+1))
    if i <= 9:
        # calculate the average difference between two
        # calculate the average consumption grouped by different temperatures
        df_dVSt = df_tou1h_hPrice[df_tou1h_hPrice.GMT.str.contains('0' + str(i) + ':00:00')] #TOU demand vs t
emperature for high price periods
        df_dVSt = df_dVSt.groupby('TempC')['Total'].mean() / (df_dVSt.shape[1] - 4)
        df_dVSt = df_dVSt.reset_index()
        ax_Ntou[-1].plot(df_dVSt.TempC, df_dVSt.Total,  c = 'green', label = 'TOU High Price', linestyle='das
hed' )

        df_NdVSt_help = df_Ntou1h[df_tariff_1h.Price == 0.672]
        df_NdVSt = df_NdVSt_help[df_NdVSt_help.GMT.str.contains('0' + str(i) + ':00:00')] # estimation of TOU
group facing default price using non-TOU data
        df_NdVSt = df_NdVSt.groupby('TempC')['Total'].mean() / (df_NdVSt.shape[1] - 4)
        df_NdVSt = df_NdVSt.reset_index()
        ax_Ntou[-1].plot(df_NdVSt.TempC, df_NdVSt.Total - const_shift[i], c = 'red', label = 'non-TOU Estimat
ion', linestyle='dashed' )

        ax_Ntou[-1].set_title('Time: 0' + str(i) + ':00')
        ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_Ntou[-1].set_ylabel('Average Consumption (kWh)')
        ax_Ntou[-1].legend()
    else:
        df_dVSt = df_tou1h_hPrice[df_tou1h_hPrice.GMT.str.contains(str(i) + ':00:00')] #TOU demand vs tempera
ture for high price periods
        df_dVSt = df_dVSt.groupby('TempC')['Total'].mean() / (df_dVSt.shape[1] - 4)
        df_dVSt = df_dVSt.reset_index()
        ax_Ntou[-1].plot(df_dVSt.TempC, df_dVSt.Total,  c = 'green', label = 'TOU High Price', linestyle='das
hed')

        df_NdVSt_help = df_Ntou1h[df_tariff_1h.Price == 0.672]
        df_NdVSt = df_NdVSt_help[df_NdVSt_help.GMT.str.contains(str(i) + ':00:00')] # estimation of TOU group
facing default price using non-TOU data
        df_NdVSt = df_NdVSt.groupby('TempC')['Total'].mean() / (df_NdVSt.shape[1] - 4)
        df_NdVSt = df_NdVSt.reset_index()
        ax_Ntou[-1].plot(df_NdVSt.TempC, df_NdVSt.Total - const_shift[i], c = 'red', label = 'non-TOU Estimat
ion', linestyle='dashed' )

        ax_Ntou[-1].set_title('Time: ' + str(i) + ':00')
        ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_Ntou[-1].set_ylabel('Average Consumption (kWh)')
        ax_Ntou[-1].legend()
plt.tight_layout()
```

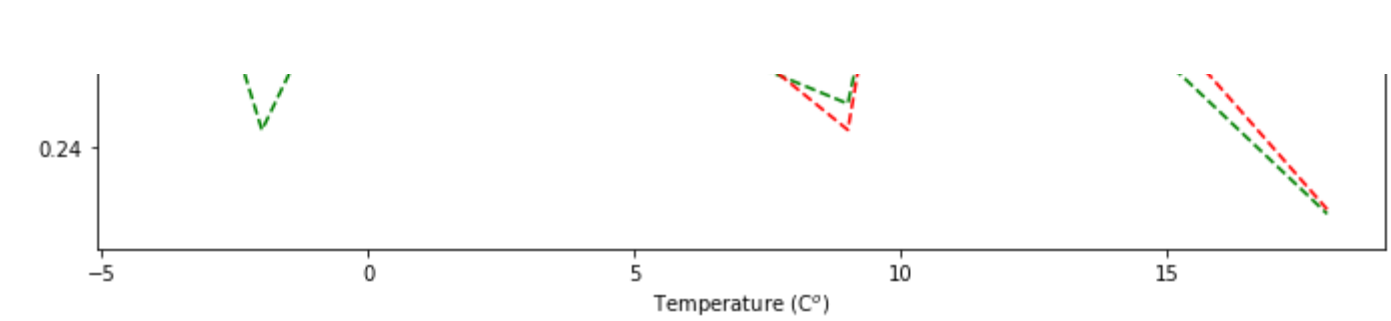

Time: 00:00
TOU High Price
non-TOU Estimation
Average Consumption (kWh)
Temperature (C°)
Time: 01:00
TOU High Price
non-TOU Estimation
Average Consumption (kWh)
Temperature (C°)
Time: 02:00
TOU High Price
non-TOU Estimation
Average Consumption (kWh)
Temperature (C°)
Time: 03:00
TOU High Price
non-TOU Estimation

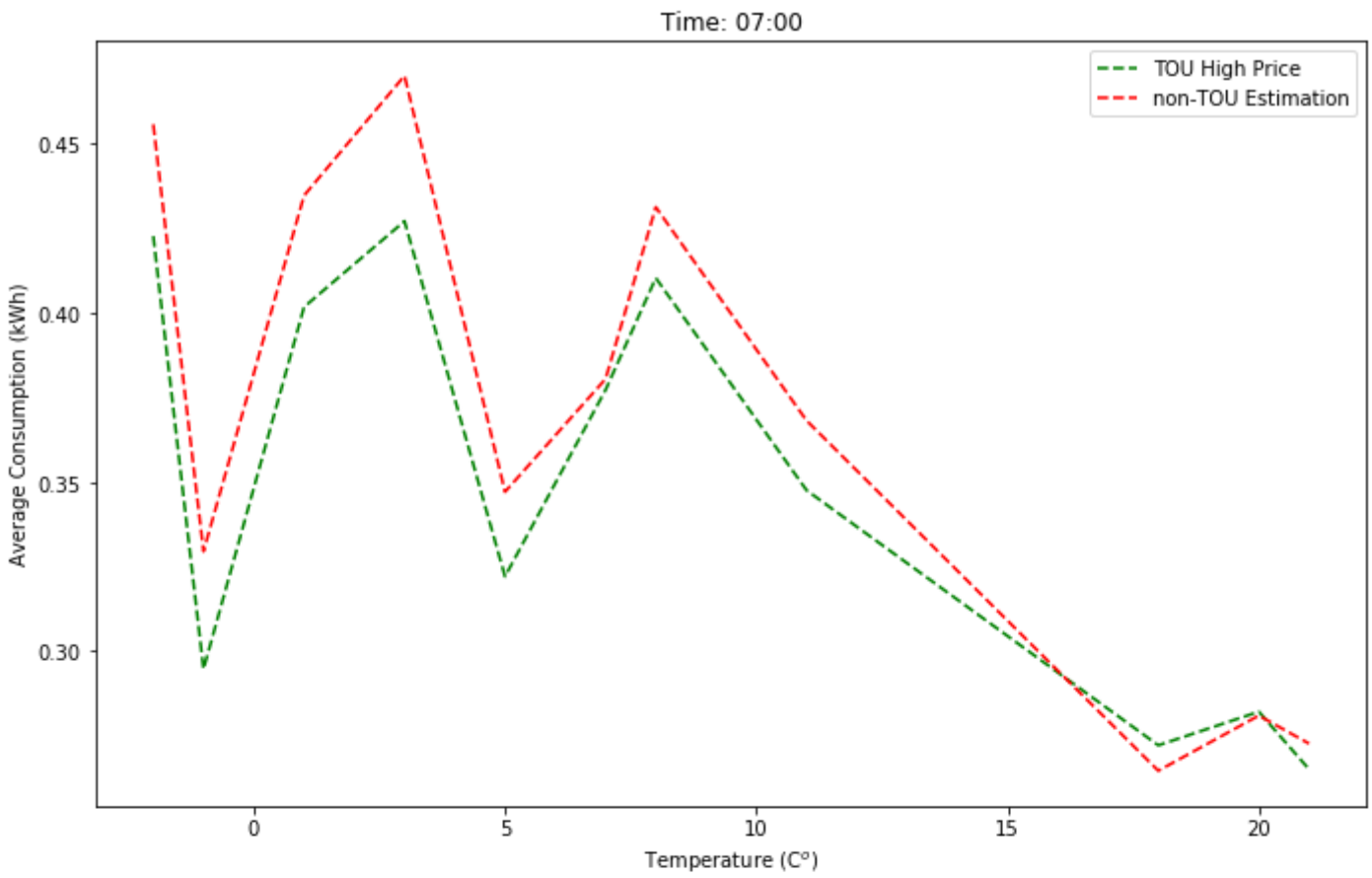

Average Consumption (kW
0.21
0.20
0.19
0.18
0.17
0
5
10
15
20
25
Temperature (C°)

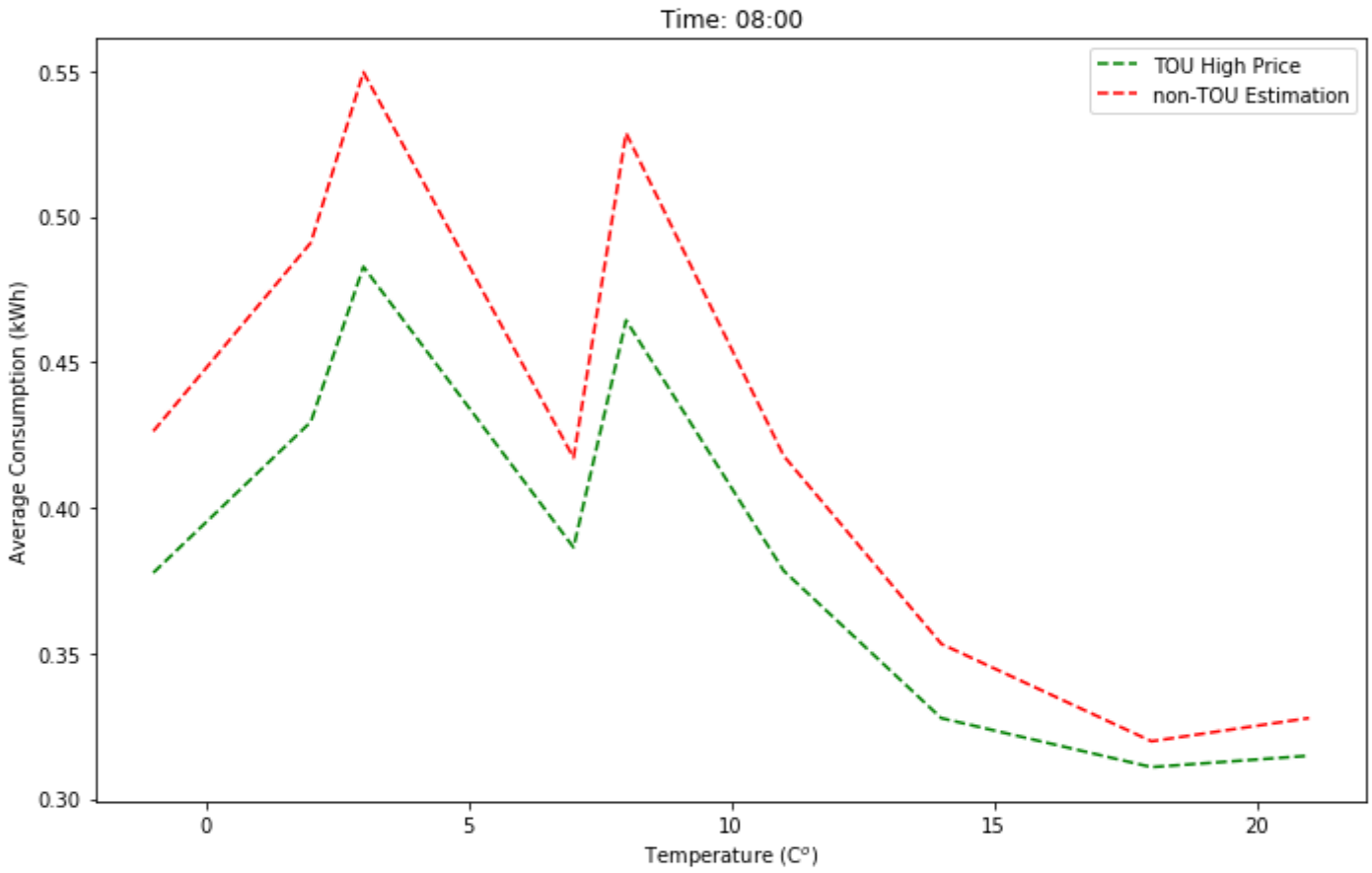

Time: 04:00
TOU High Price
non-TOU Estimation
Average Consumption (kWh)
0.23
0.22
0.21
0.20
0.19
0.18
2
4
6
8
10
12
14
16
Temperature (C°)

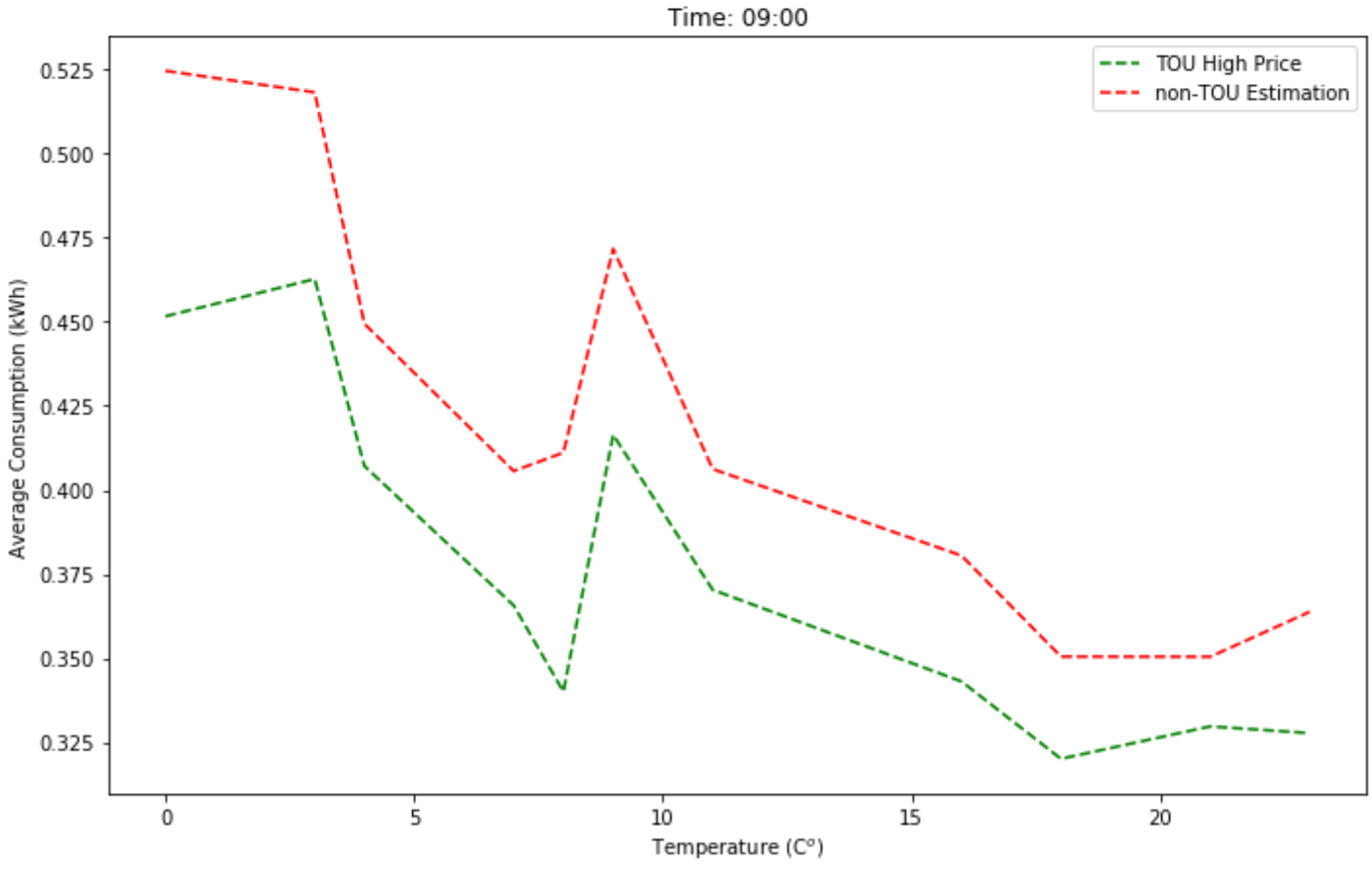

Time: 05:00
TOU High Price
non-TOU Estimation
Average Consumption (kWh)
0.26
0.25
0.24
0.23
0.22
0.21
0.20
−5
0
5
10
15
Temperature (C°)

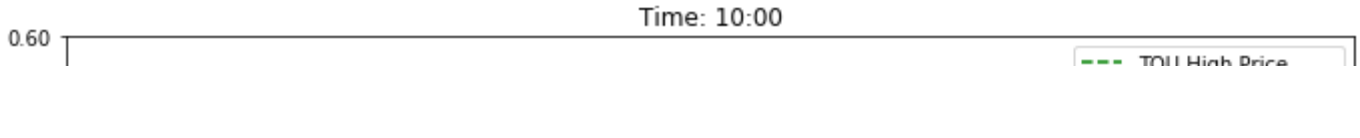
Time: 06:00
TOU High Price
non-TOU Estimation
Average Consumption (kWh)
0.36
0.34
0.32
0.30
0.28
0.26

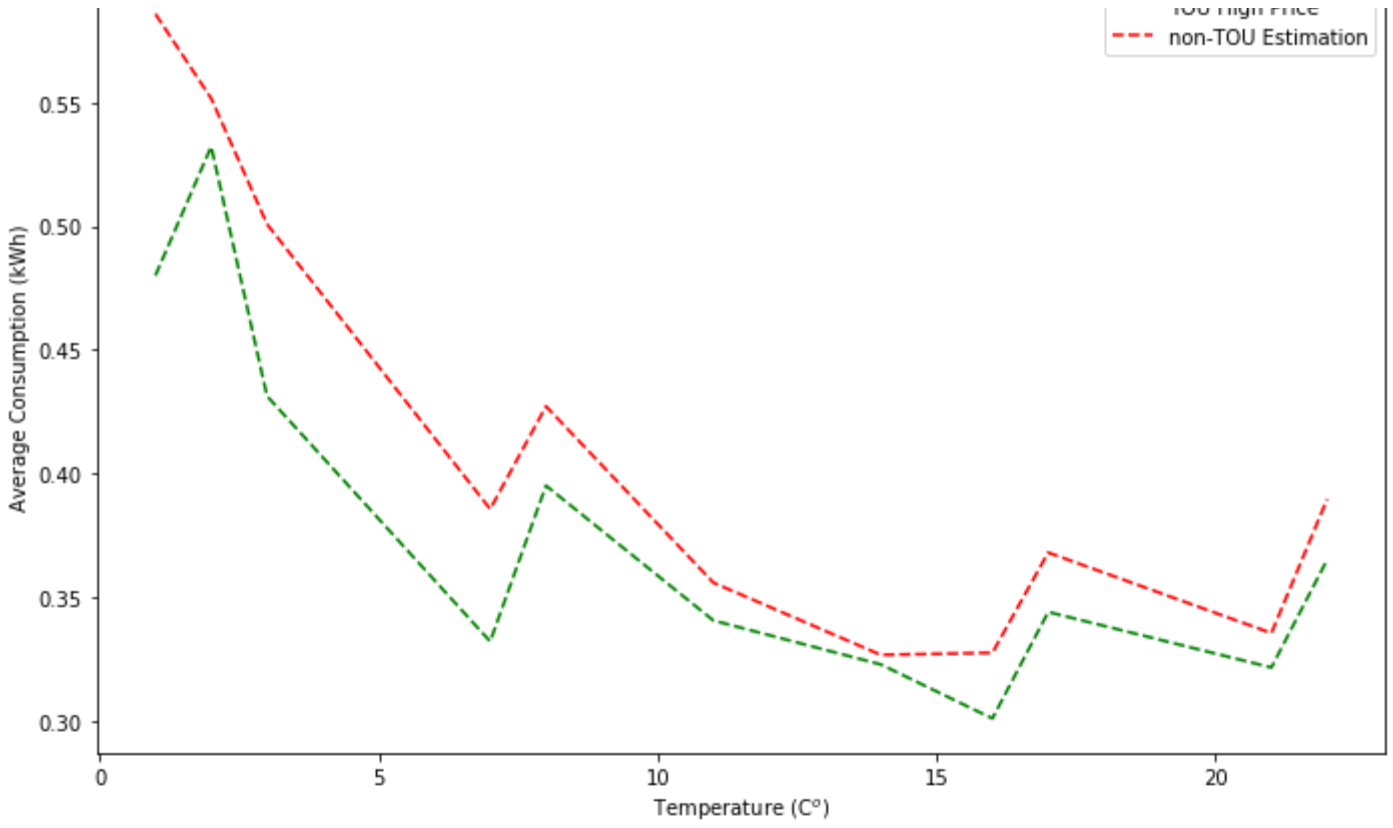

0.24
−5
0
5
10
15
Temperature (C°)


Time: 07:00
TOU High Price
non-TOU Estimation
Average Consumption (kWh)
0.45
0.40
0.35
0.30
0
5
10
15
20
Temperature (C°)

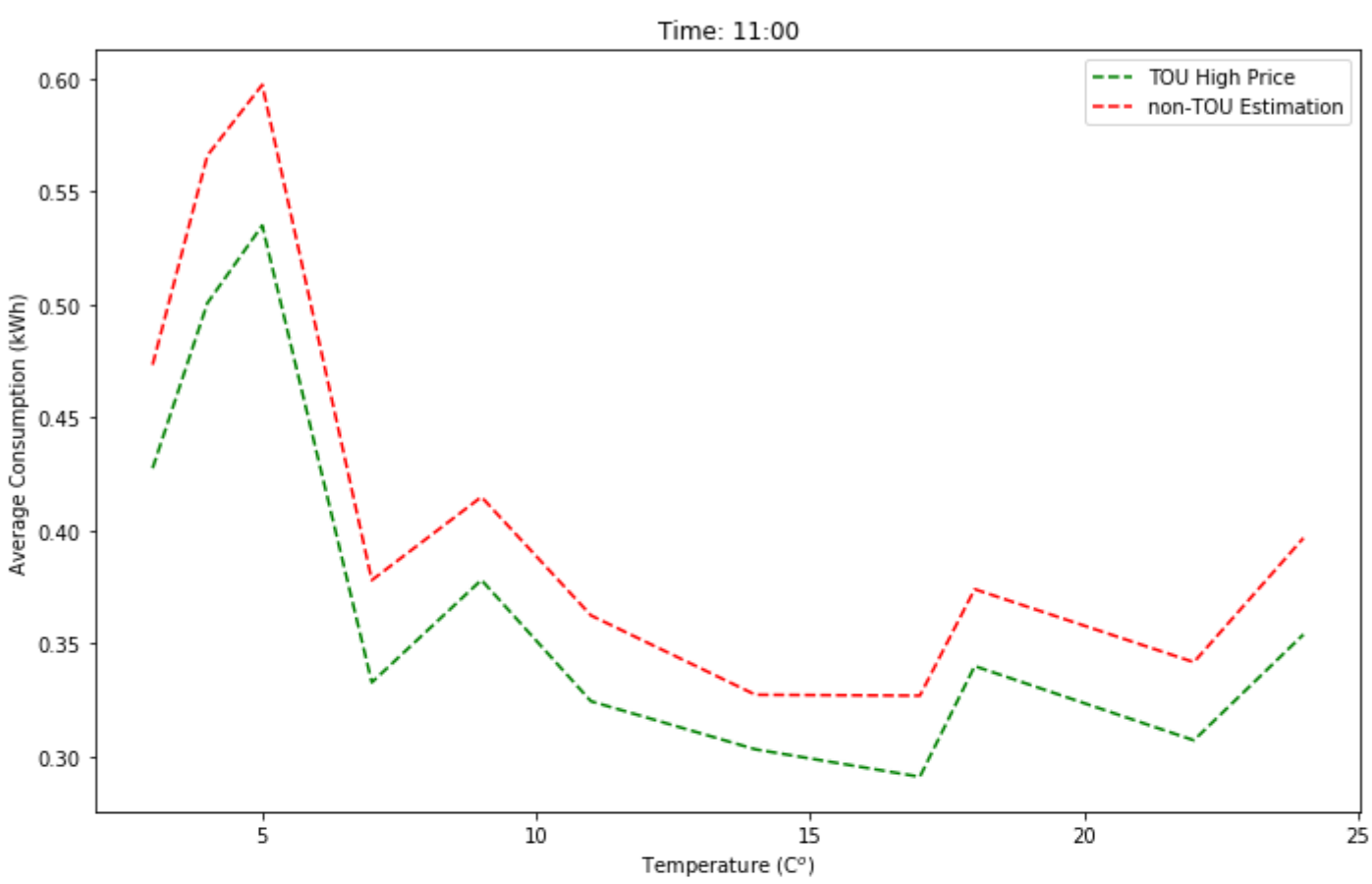

Time: 08:00
TOU High Price
non-TOU Estimation
Average Consumption (kWh)
0.55
0.50
0.45
0.40
0.35
0.30
0
5
10
15
20
Temperature (C°)


Time: 09:00
TOU High Price
non-TOU Estimation
Average Consumption (kWh)
0.525
0.500
0.475
0.450
0.425
0.400
0.375
0.350
0.325
0
5
10
15
20
Temperature (C°)

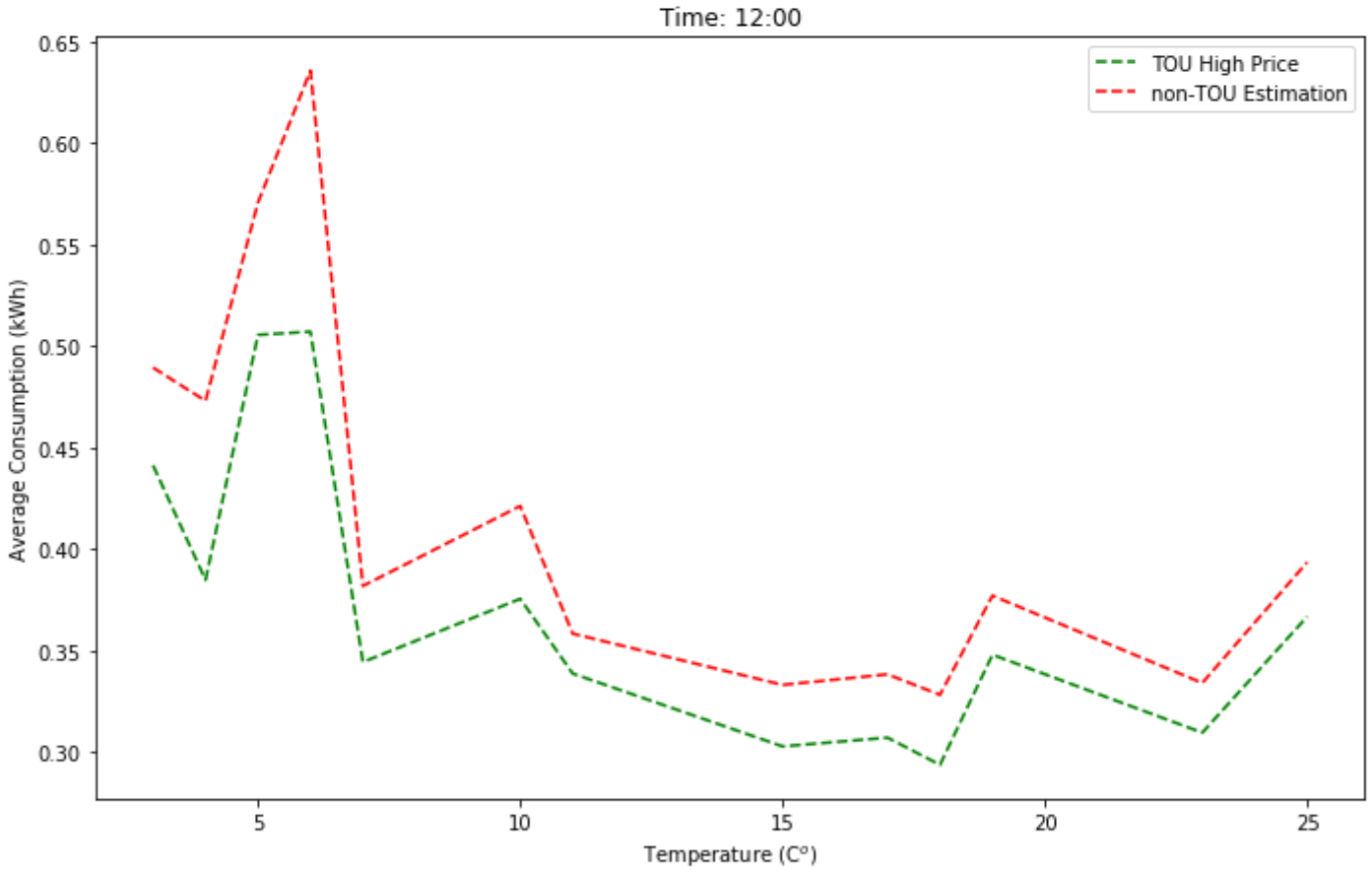

Time: 10:00
0.60



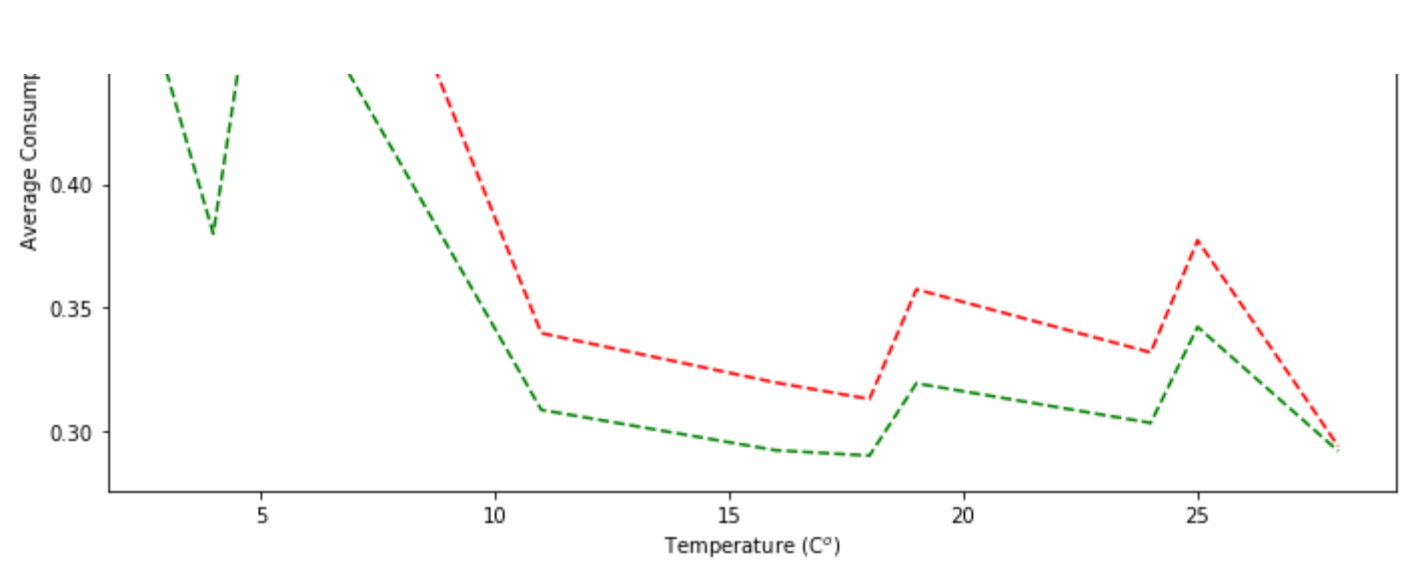
Time: 11:00

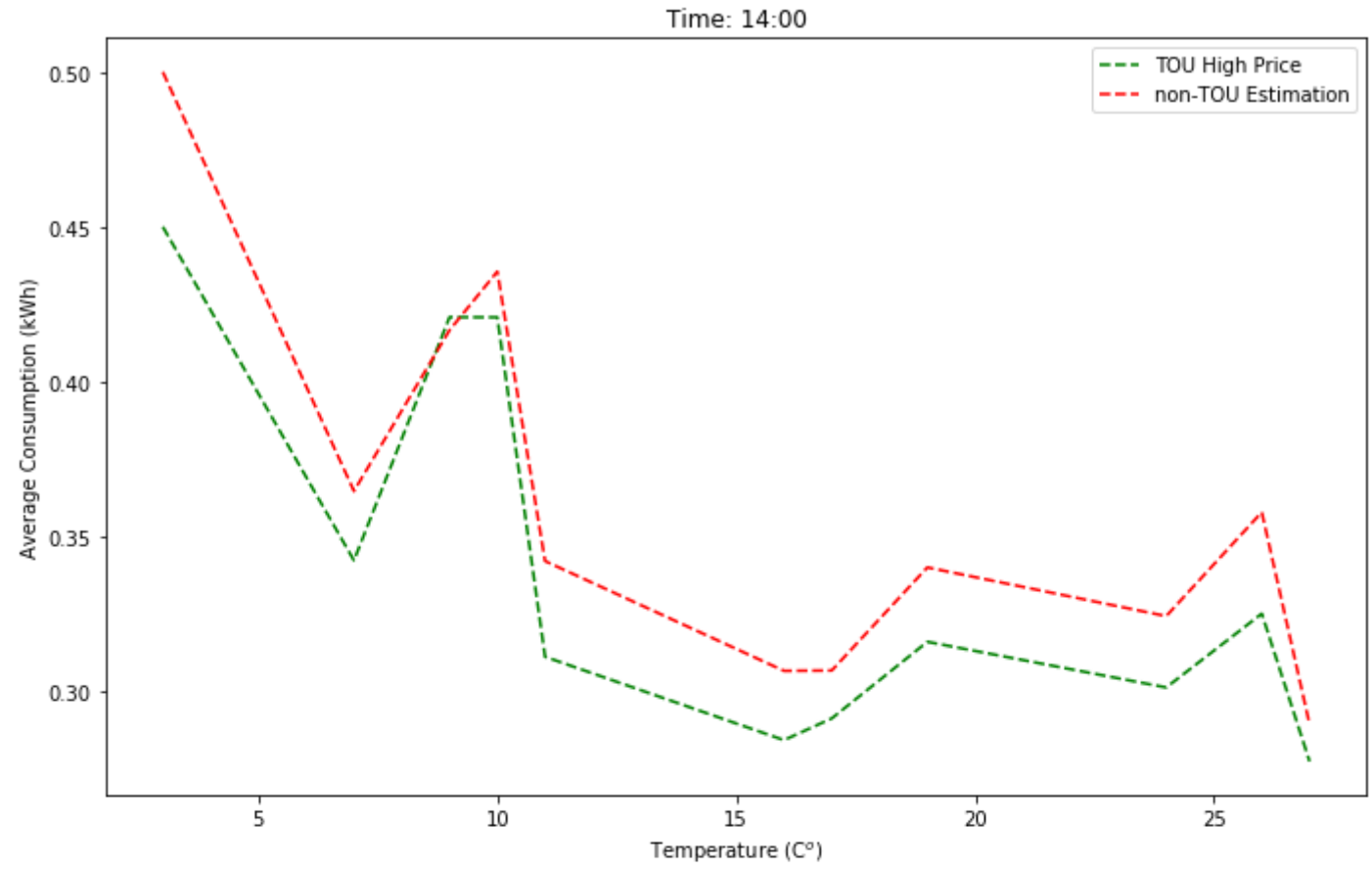


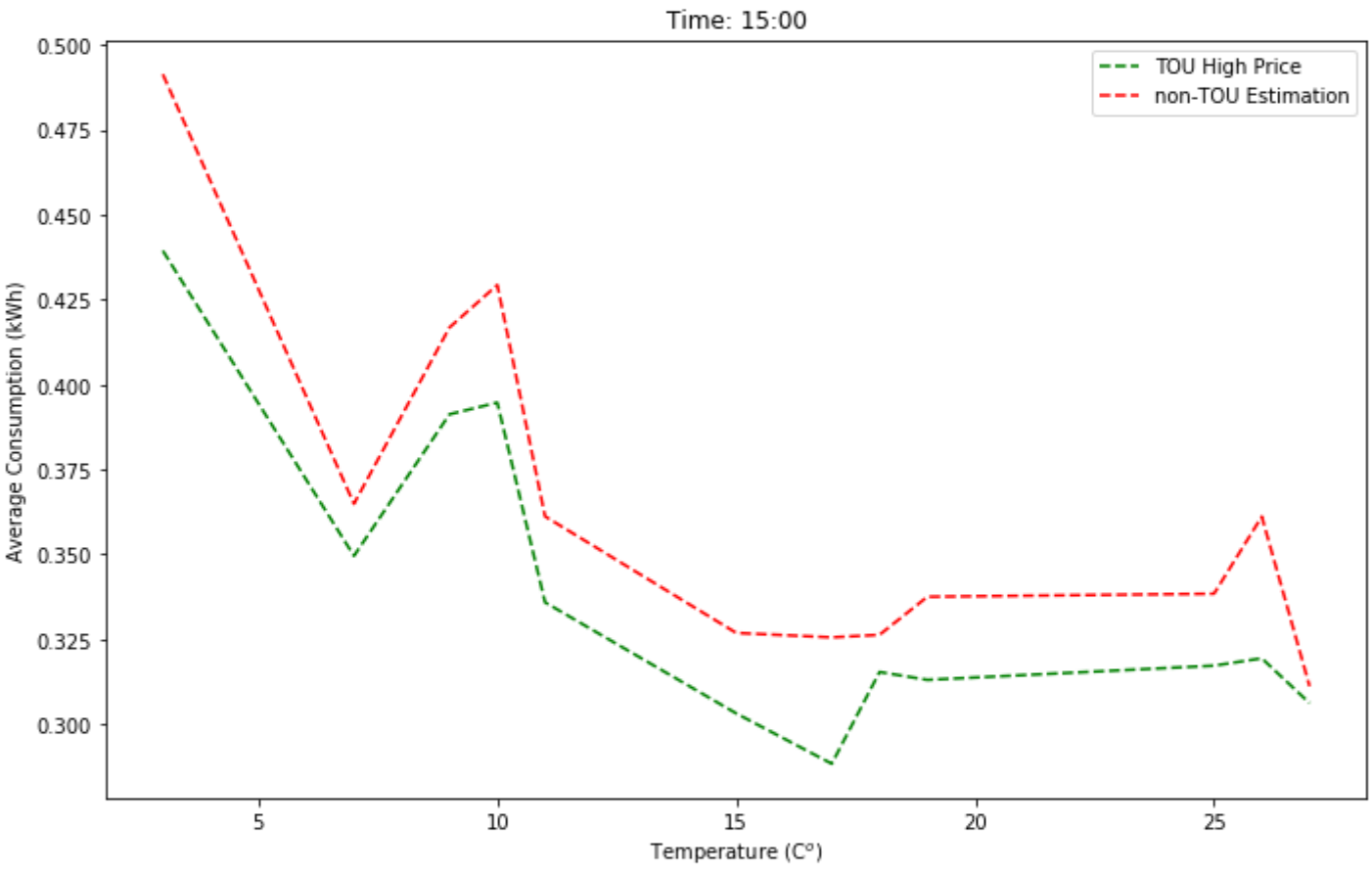
Time: 12:00



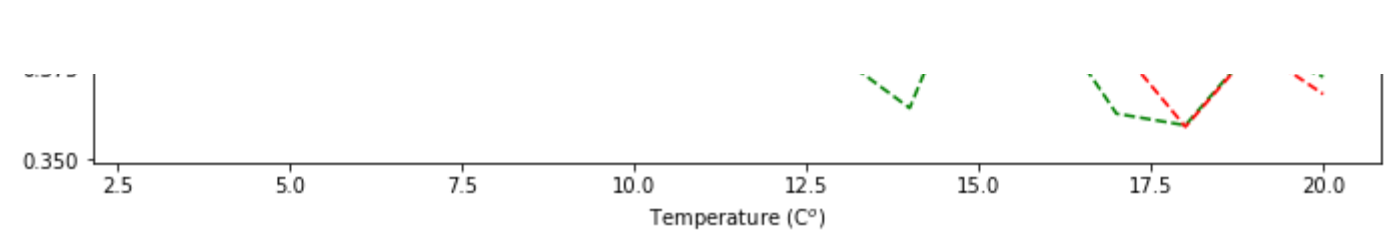
Time: 13:00

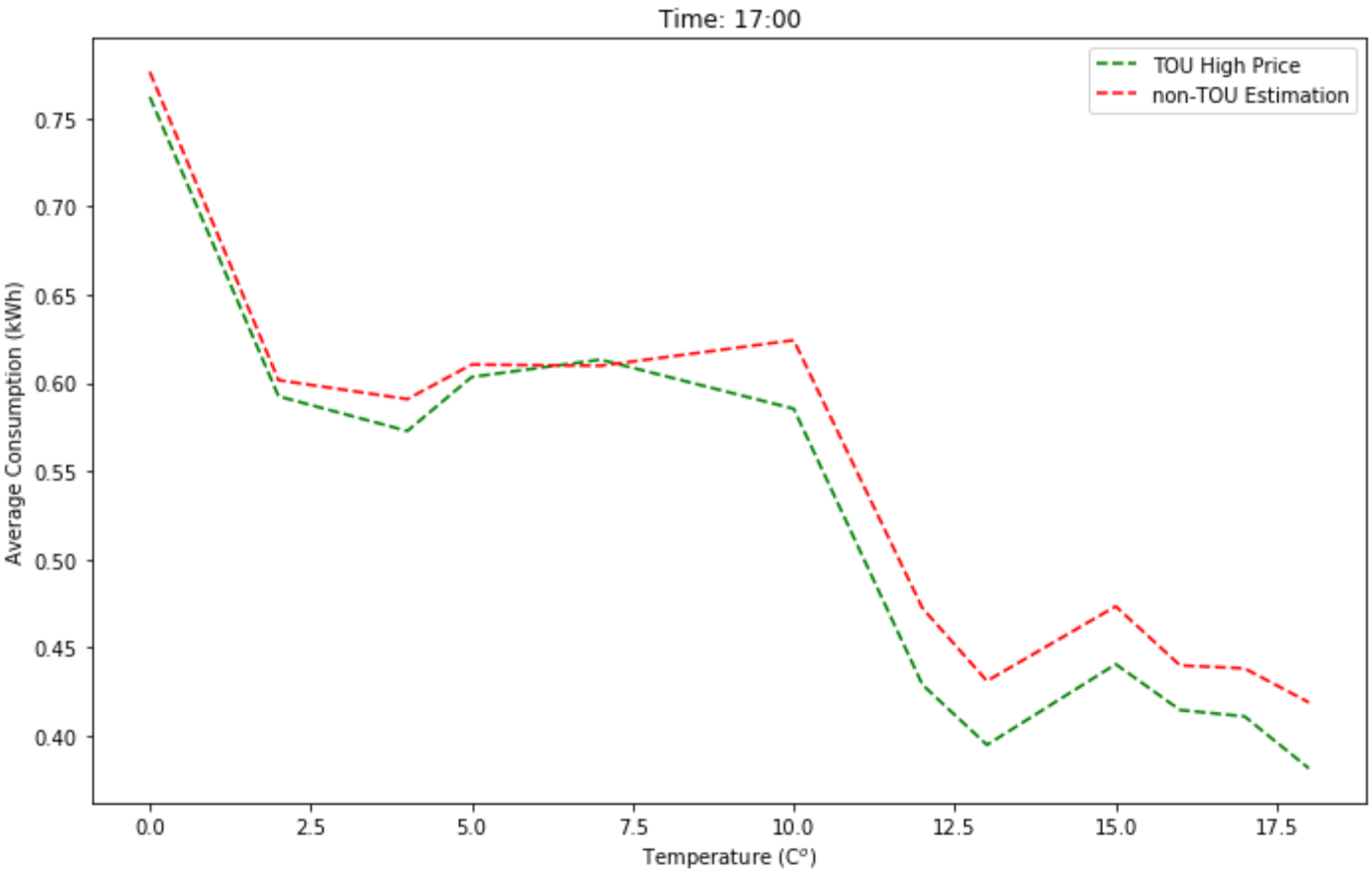

0.40
0.35
0.30
5
10
15
20
25
Temperature (C°)

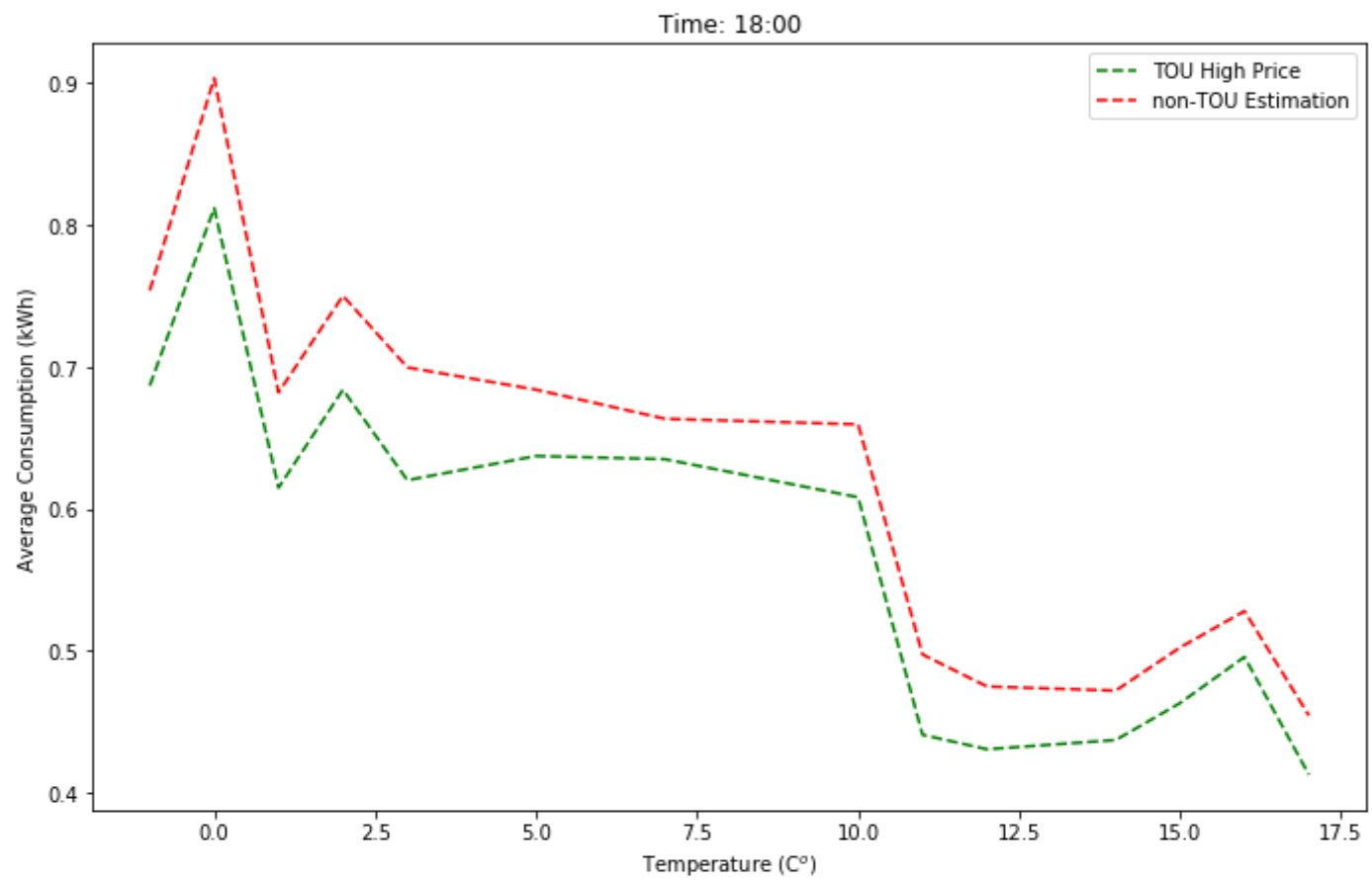

Time: 14:00
TOU High Price
non-TOU Estimation
0.50
0.45
0.40
0.35
0.30
5
10
15
20
25
Temperature (C°)

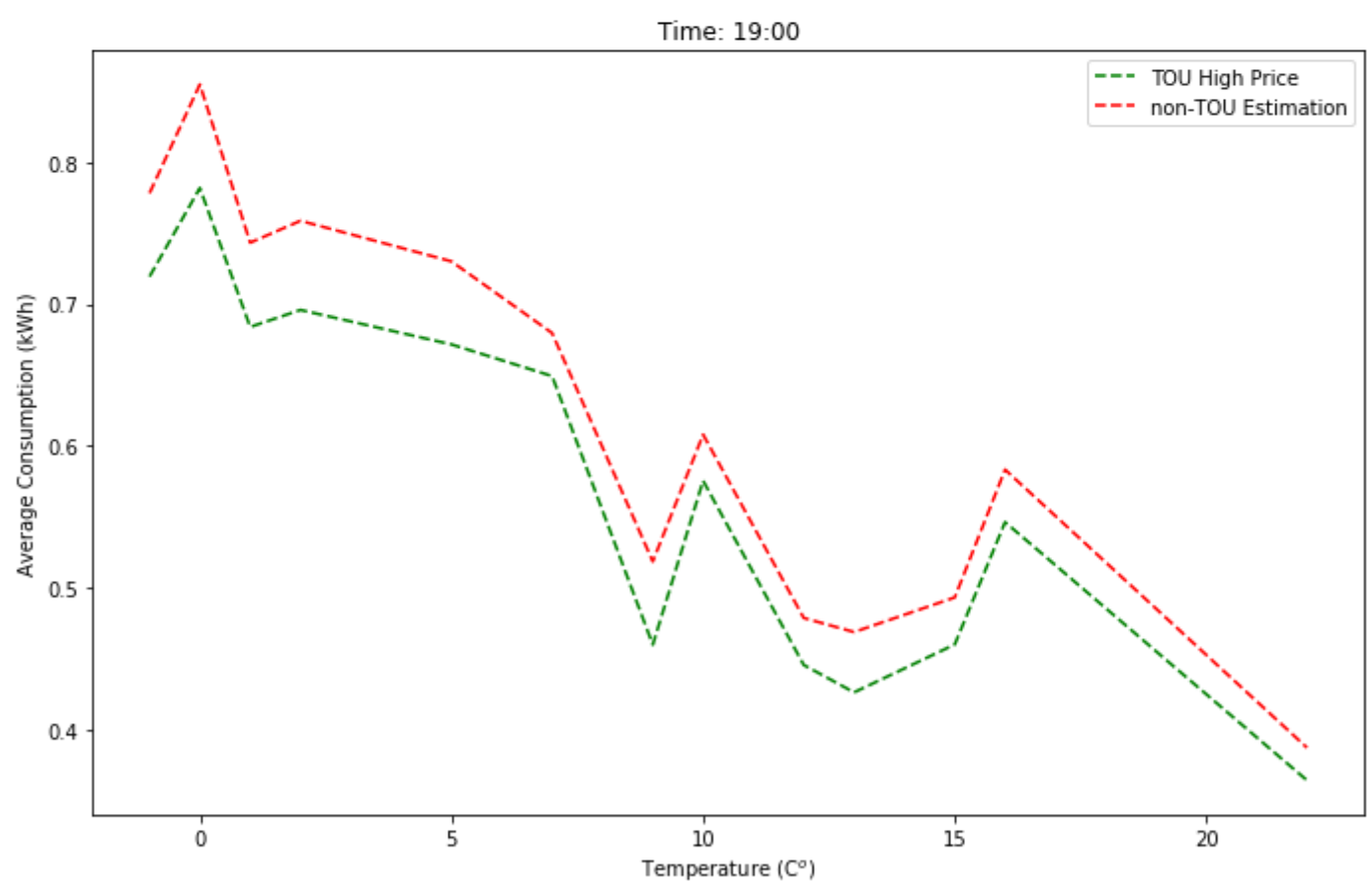

Time: 15:00
TOU High Price
non-TOU Estimation
Average Consumption (kWh)
0.500
0.475
0.450
0.425
0.400
0.375
0.350
0.325
0.300
5
10
15
20
25
Temperature (C°)

Time: 16:00
TOU High Price
non-TOU Estimation
Average Consumption (kWh)
0.550
0.525
0.500
0.475
0.450
0.425
0.400
0.375

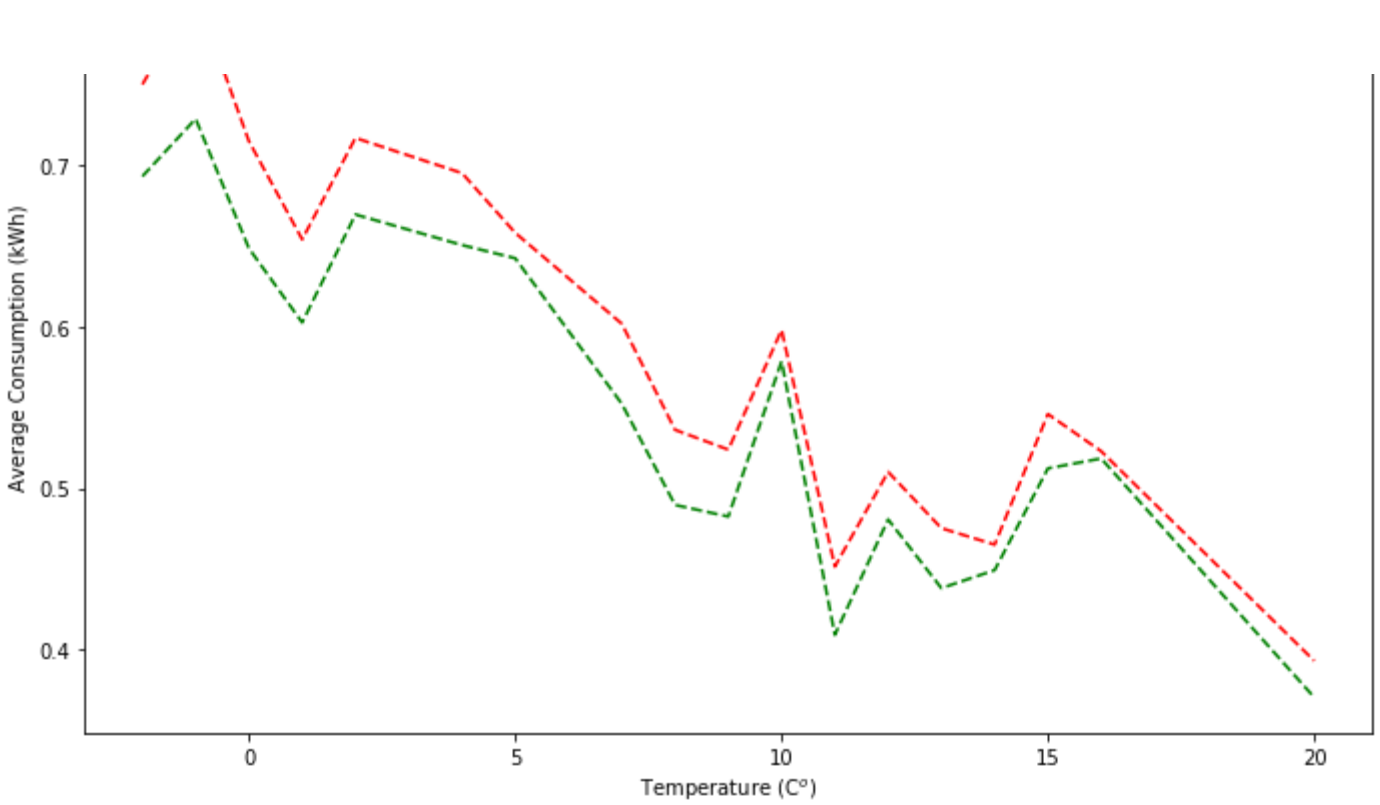

0.350
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Temperature (C°)

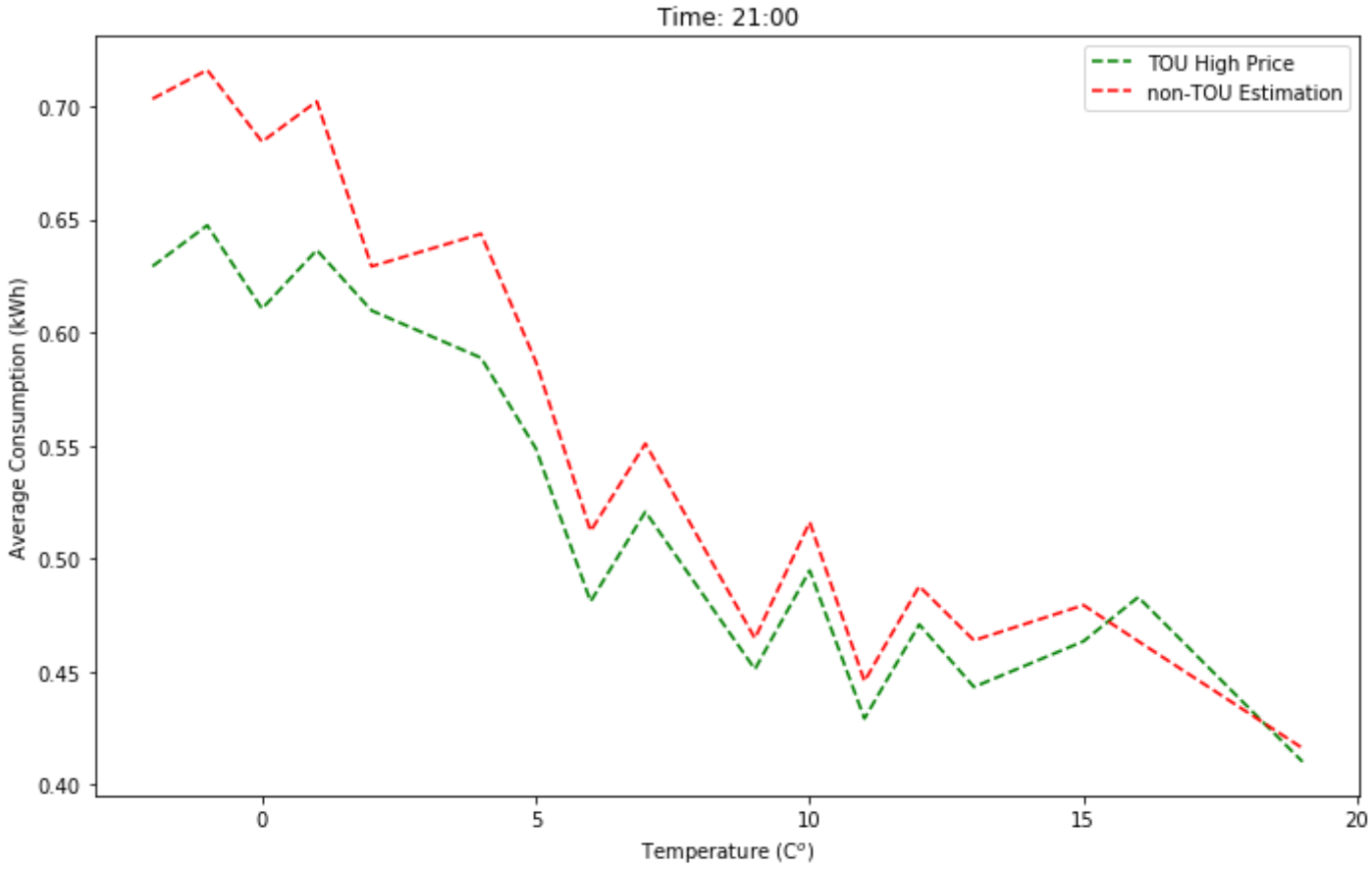

Time: 17:00
TOU High Price
non-TOU Estimation
Average Consumption (kWh)
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Temperature (C°)

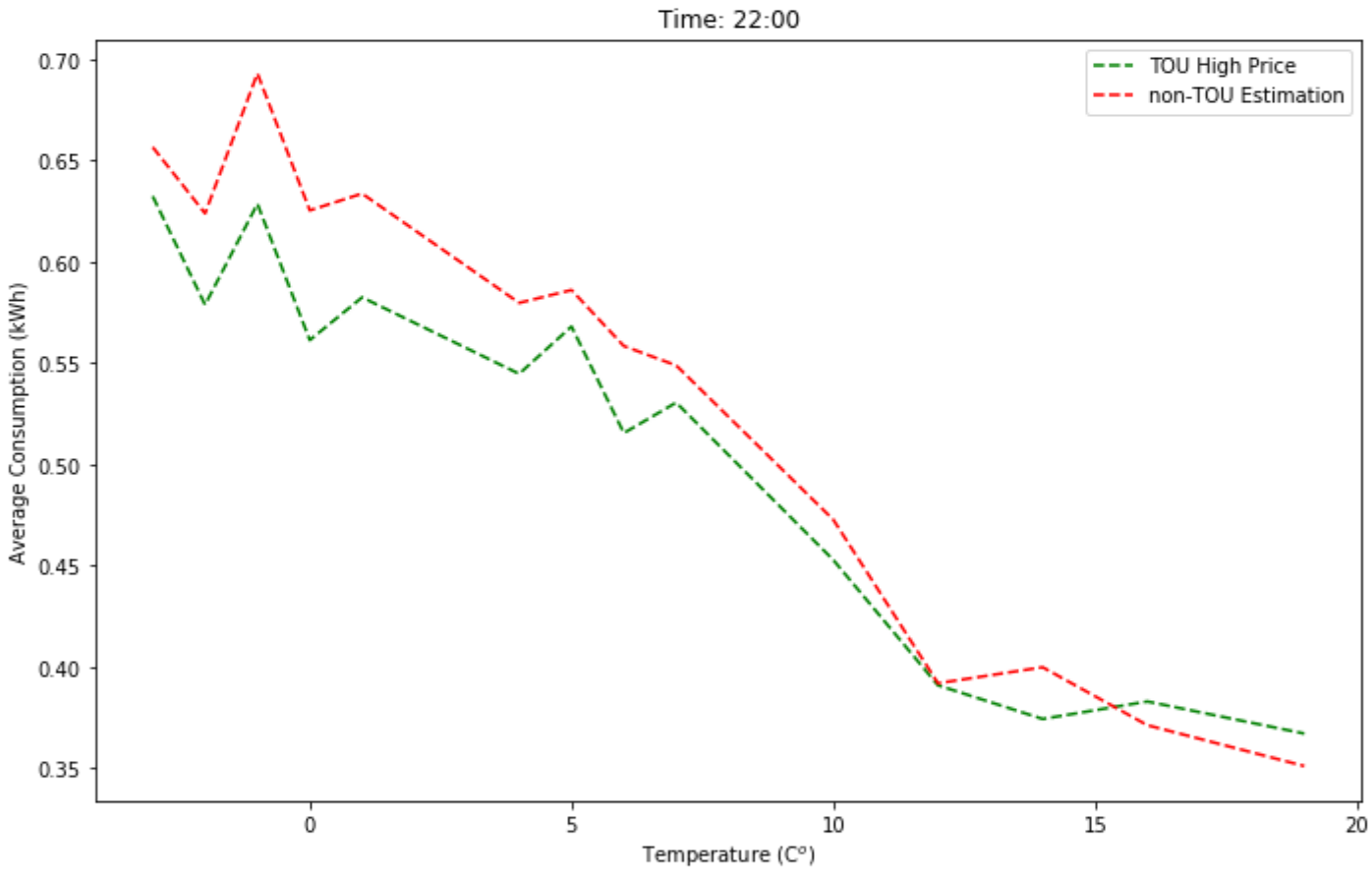

Time: 18:00
TOU High Price
non-TOU Estimation
Average Consumption (kWh)
0.9
0.8
0.7
0.6
0.5
0.4
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Temperature (C°)

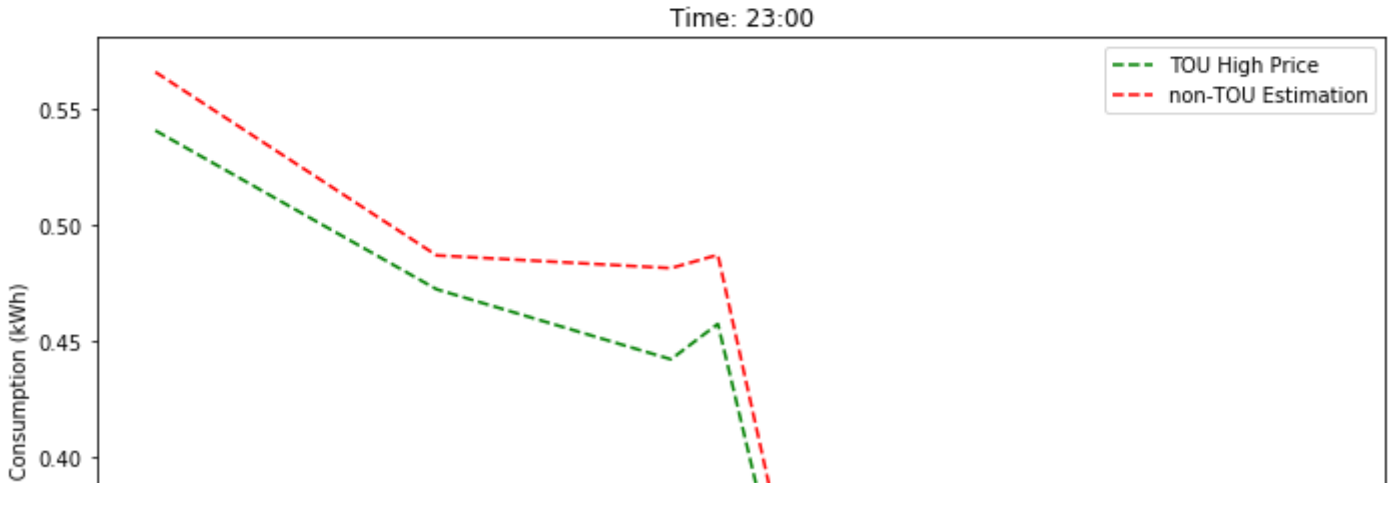

Time: 19:00
TOU High Price
non-TOU Estimation
Average Consumption (kWh)
0.8
0.7
0.6
0.5
0.4
0
5
10
15
20
Temperature (C°)

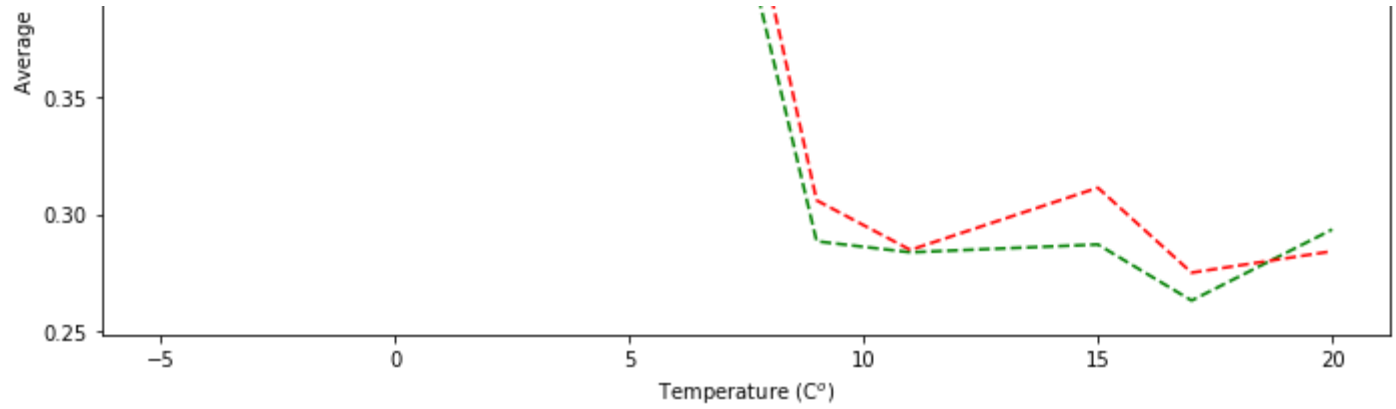

Time: 20:00
TOU High Price
non-TOU Estimation
0.8


0.7
0.6
0.5
0.4
Average Consumption (kWh)
0
5
10
15
20
Temperature (C°)


Time: 21:00
TOU High Price
non-TOU Estimation
0.70
0.65
0.60
0.55
0.50
0.45
0.40
Average Consumption (kWh)
0
5
10
15
20
Temperature (C°)


Time: 22:00
TOU High Price
non-TOU Estimation
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0
5
10
15
20
Temperature (C°)


Time: 23:00
TOU High Price
non-TOU Estimation
0.55
0.50
0.45
0.40
Consumption (kWh)



In [108]:
```
# High price periods high 0.672 vs. default 0.1176 and low 0.0399
# Note: CM events only use high and low prices, so when we pick the high price and low price periods,
# those in CM will also be picked.
# first, create a list of days that belongs to event days
h_event_time = set()
df_event = df_tariff_1h[df_tariff_1h.Event_tags.notnull()]
# df_Ntou1h_hPrice = df_Ntou1h[df_tariff_1h.Price == 0.672]
# df_Ntou1h_lPrice = df_Ntou1h[df_tariff_1h.Price == 0.0399]
df_tou1h_hPrice = df_tou1h[df_tariff_1h.Price == 0.672]
df_tou1h_lPrice = df_tou1h[df_tariff_1h.Price == 0.0399]

# All event day's dataframe
df_help_event = df_tariff_1h[df_help['GMT'].isin(event_days)]
# the dataframe in event days with default price
df_help_event = df_help_event[df_help_event.Price == 0.1176]
# TOU dataframe corresponding to these default price in event days
df_tou1h_eventday_dPrice = df_tou1h.loc[df_help_event.index]
```

In [114]:
```
df_NdVSt_help = df_Ntou1h[df_help['GMT'].isin(event_days)]
df_NdVSt_help
```

Out[114]:

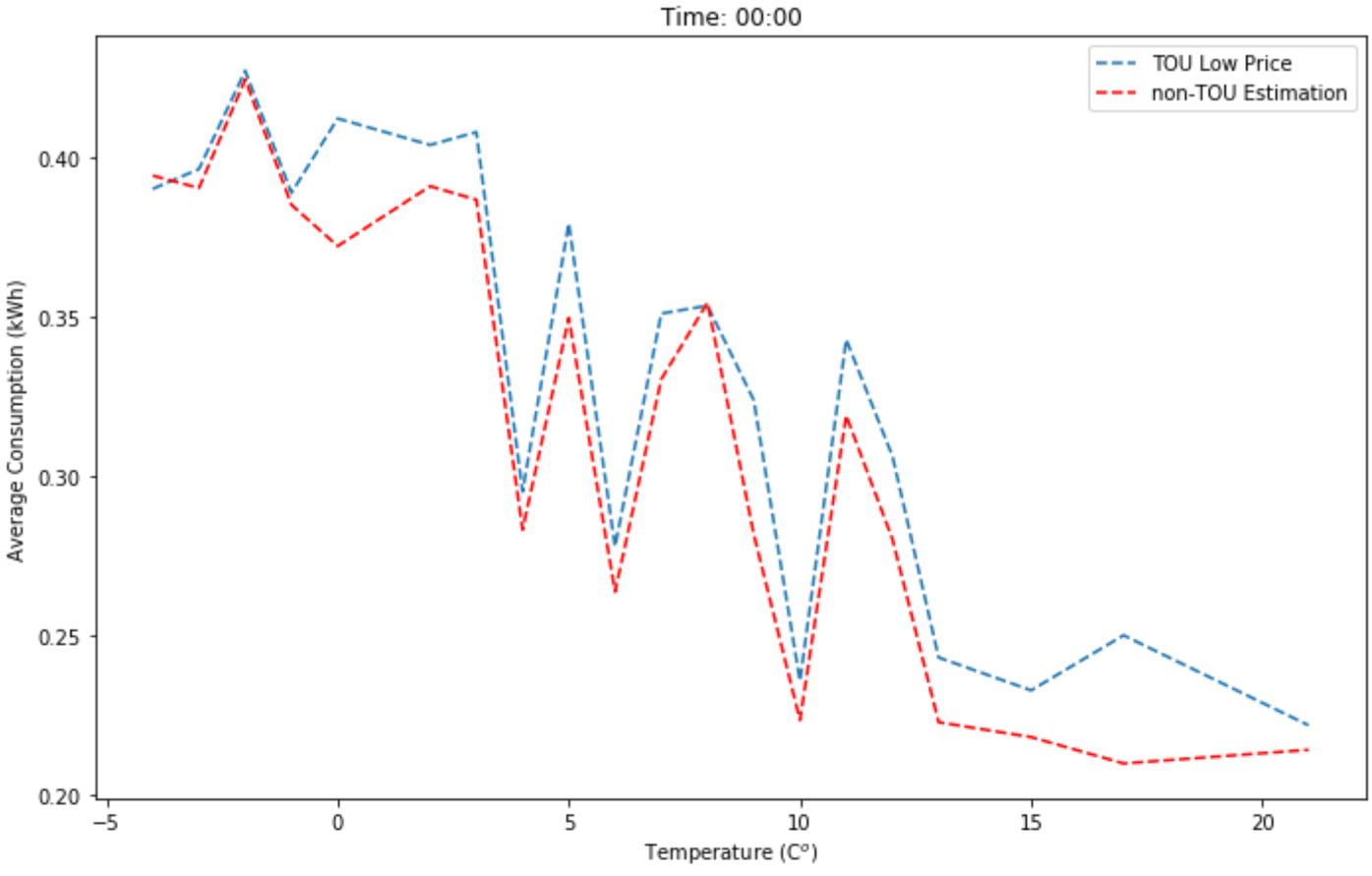
	GMT	N0000	N0001	N0002	N0003	N0004	N0005	N0006	N0007	N0008	...	N4166	N4167	N4168	N4169	N4170	N4171	N4
72	2013-01-04 00:00:00	0.354	0.265	0.083	0.097	0.634	0.125	0.000	0.169	0.116	...	0.037	0.352	0.189	0.173	0.168	0.020	0.3
73	2013-01-04 01:00:00	0.296	0.231	0.097	0.089	0.595	0.079	0.000	0.151	0.146	...	0.059	0.127	0.193	0.109	0.154	0.023	0.1
74	2013-01-04 02:00:00	0.303	0.263	0.043	0.115	0.408	0.072	0.000	0.169	0.144	...	0.020	0.207	0.199	0.087	0.086	0.100	0.1
75	2013-01-04 03:00:00	0.362	0.266	0.032	0.143	0.158	0.074	0.000	0.151	0.130	...	0.021	0.131	0.190	0.078	0.144	0.114	0.2
76	2013-01-04 04:00:00	0.326	0.243	0.106	0.059	0.136	0.080	0.000	0.169	0.131	...	0.075	0.250	0.173	0.078	0.108	0.112	0.5
77	2013-01-04 05:00:00	0.345	0.252	0.091	0.054	0.116	0.102	0.000	0.152	0.117	...	0.773	0.140	0.184	0.107	0.120	0.110	0.1
78	2013-01-04 06:00:00	0.650	0.263	0.030	0.070	0.116	0.395	0.000	0.168	0.153	...	2.840	1.492	0.211	0.079	0.140	0.109	0.1
79	2013-01-04 07:00:00	0.605	0.250	0.041	0.507	0.116	0.361	0.000	0.150	0.185	...	4.225	0.909	0.215	0.078	0.090	0.135	0.3
80	2013-01-04 08:00:00	0.336	0.242	0.652	1.307	0.118	0.329	0.000	0.166	0.449	...	2.781	1.191	0.319	0.090	0.142	0.287	0.0
81	2013-01-04 09:00:00	0.318	0.264	0.615	0.920	0.571	0.669	0.000	0.151	0.978	...	0.993	0.701	0.370	0.175	0.074	0.108	0.0
82	2013-01-04 10:00:00	0.326	0.259	0.232	0.841	0.302	0.071	0.000	0.162	0.800	...	0.044	1.587	0.197	0.112	0.140	0.107	0.0
83	2013-01-04 11:00:00	0.351	0.233	0.314	0.148	0.216	0.071	0.000	0.153	0.170	...	0.394	2.544	0.145	0.087	0.266	0.107	0.0
84	2013-01-04 12:00:00	0.281	0.261	0.323	0.131	1.112	0.073	0.000	0.160	0.190	...	0.726	1.254	0.083	0.081	0.361	0.105	0.0
85	2013-01-04 13:00:00	0.880	0.261	0.281	0.157	0.638	0.381	0.000	0.156	0.273	...	0.024	0.378	0.099	0.078	0.218	0.048	0.0
86	2013-01-04 14:00:00	0.622	0.227	0.160	0.215	0.251	1.204	0.000	0.159	0.838	...	0.074	0.499	0.180	0.108	0.865	0.019	0.0
87	2013-01-04 15:00:00	0.343	0.263	0.359	0.216	0.136	0.745	0.000	0.158	0.901	...	0.070	0.309	0.458	0.078	0.387	0.310	0.3
88	2013-01-04 16:00:00	0.391	0.264	0.284	0.572	0.114	1.712	0.000	0.156	1.027	...	0.800	0.550	0.273	0.077	0.456	0.228	0.7
89	2013-01-04 17:00:00	0.414	0.230	0.758	0.570	0.210	0.485	0.000	0.155	0.676	...	2.230	0.474	0.328	0.076	0.303	0.394	0.4
90	2013-01-04 18:00:00	0.590	0.260	0.577	0.662	0.586	0.554	0.000	0.197	0.550	...	4.168	0.660	0.735	0.112	0.302	0.113	0.4
91	2013-01-04 19:00:00	0.847	0.266	0.372	1.363	0.907	0.502	0.000	0.215	0.637	...	4.791	0.500	1.085	0.447	0.315	0.107	0.6
92	2013-01-04 20:00:00	0.542	0.243	0.425	0.876	1.155	0.761	0.000	0.199	1.849	...	4.168	0.607	0.337	0.382	0.582	0.108	0.3

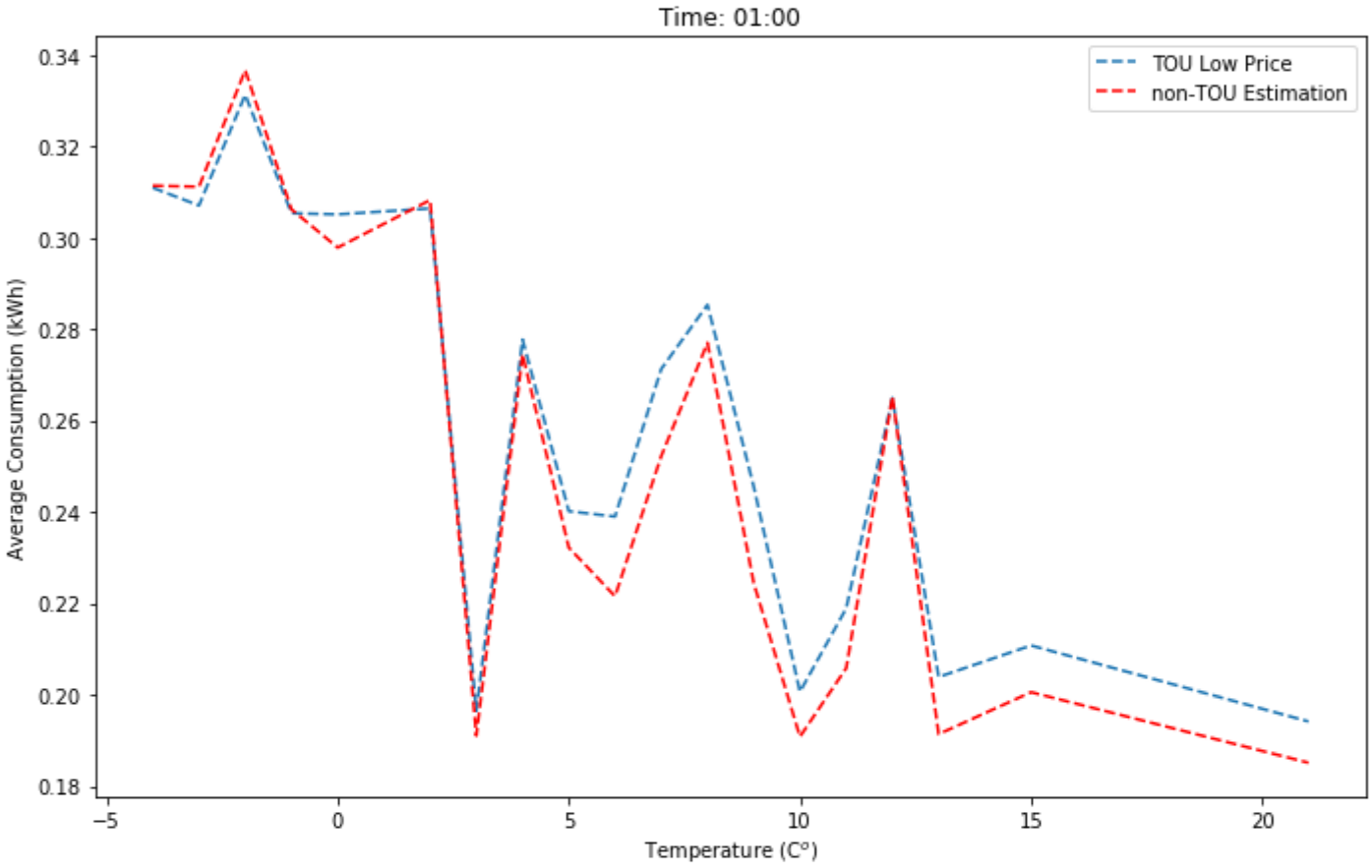
	GMT	N0000	N0001	N0002	N0003	N0004	N0005	N0006	N0007	N0008	...	N4166	N4167	N4168	N4169	N4170	N4171	N4
93	2013-01-04 21:00:00	1.009	0.246	0.079	0.593	0.715	0.406	0.000	0.218	0.616	...	3.137	2.029	0.301	0.415	0.329	0.107	0.4
94	2013-01-04 22:00:00	0.423	0.264	0.032	0.475	0.556	0.602	0.000	0.216	0.612	...	0.792	0.689	0.324	0.519	0.441	0.139	0.3
95	2013-01-04 23:00:00	0.408	0.255	0.094	0.189	0.354	0.269	0.000	0.198	0.236	...	0.328	0.392	0.212	0.287	0.292	0.138	0.3
144	2013-01-07 00:00:00	0.375	0.259	0.079	0.108	0.375	0.153	0.000	0.156	0.131	...	0.062	0.326	0.218	0.124	0.147	0.112	0.1
145	2013-01-07 01:00:00	0.324	0.266	0.037	0.157	0.268	0.088	0.000	0.158	0.156	...	0.021	0.161	0.213	0.095	0.145	0.113	0.1
146	2013-01-07 02:00:00	0.317	0.229	0.057	0.133	0.274	0.078	0.000	0.164	0.134	...	0.061	0.187	0.171	0.082	0.158	0.112	0.1
147	2013-01-07 03:00:00	0.330	0.265	0.098	0.095	0.118	0.078	0.000	0.153	0.177	...	0.033	0.151	0.185	0.094	0.107	0.110	0.6
148	2013-01-07 04:00:00	0.416	0.267	0.088	0.069	0.119	0.087	0.000	0.169	0.121	...	0.021	0.171	0.169	0.080	0.116	0.111	0.3
149	2013-01-07 05:00:00	0.603	0.229	0.031	0.054	0.236	0.082	0.000	0.149	0.166	...	0.397	0.325	0.138	0.081	0.135	0.115	0.1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
8706	2013-12-29 18:00:00	0.834	0.591	0.739	0.666	2.582	0.253	0.113	0.255	0.628	...	4.115	1.399	0.209	NaN	0.376	1.037	0.4
8707	2013-12-29 19:00:00	1.076	0.498	0.442	1.564	1.462	0.401	0.207	0.213	0.597	...	3.209	1.467	0.304	NaN	1.112	1.198	0.5
8708	2013-12-29 20:00:00	0.212	0.587	0.406	1.263	2.211	0.607	0.221	0.204	1.063	...	2.516	1.341	0.275	NaN	0.540	0.117	0.4
8709	2013-12-29 21:00:00	0.309	0.508	0.174	1.136	1.331	0.651	0.250	0.229	1.165	...	1.922	2.656	0.311	NaN	0.376	0.113	0.3
8710	2013-12-29 22:00:00	0.212	0.795	0.062	0.527	1.131	1.113	0.233	0.216	0.701	...	0.496	1.695	0.275	NaN	0.241	0.108	0.3
8711	2013-12-29 23:00:00	0.334	0.623	0.044	0.269	0.774	2.273	0.046	0.175	0.697	...	0.391	0.916	0.321	NaN	0.264	0.029	0.4
8712	2013-12-30 00:00:00	0.201	0.651	0.098	0.250	0.710	0.220	0.172	0.162	0.180	...	0.316	0.107	0.214	NaN	0.164	0.022	0.3
8713	2013-12-30 01:00:00	0.297	0.603	0.099	0.244	0.665	0.074	0.036	0.229	0.133	...	0.022	0.124	0.243	NaN	0.216	0.098	0.3
8714	2013-12-30 02:00:00	0.489	0.476	0.080	0.203	0.704	0.063	0.030	0.192	0.158	...	0.021	0.105	0.197	NaN	0.185	0.114	0.5
8715	2013-12-30 03:00:00	0.470	0.503	0.061	0.278	0.620	0.089	0.043	0.156	0.195	...	0.020	0.077	0.173	NaN	0.091	0.337	0.2
8716	2013-12-30 04:00:00	0.511	0.556	0.108	0.226	0.695	0.060	0.045	0.169	0.088	...	0.229	0.149	0.178	NaN	0.169	0.082	0.1
8717	2013-12-30 05:00:00	0.484	0.526	0.068	0.202	1.016	0.085	0.043	0.165	0.097	...	0.860	0.068	0.169	NaN	0.078	0.020	0.1

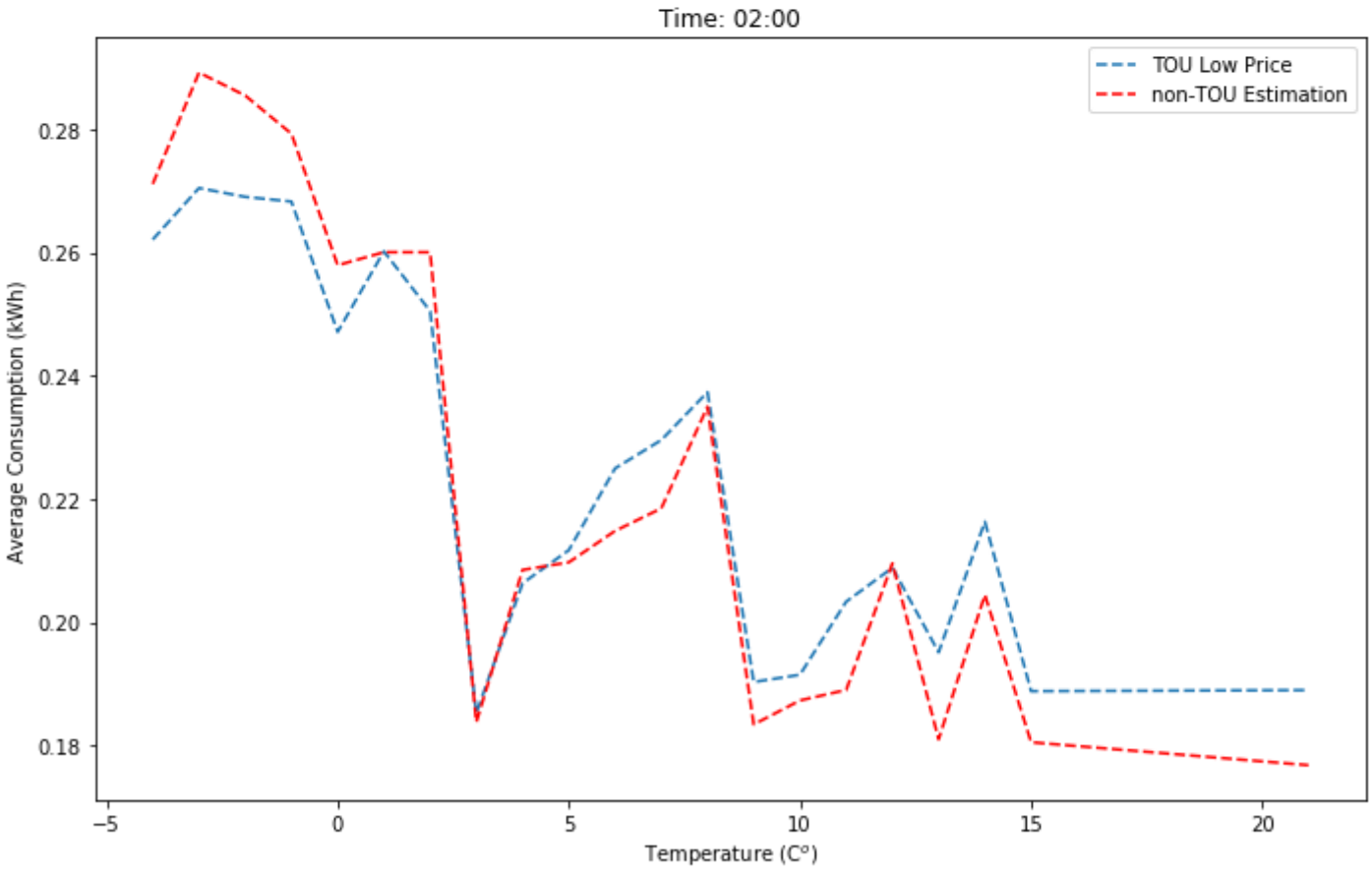
	GMT	N0000	N0001	N0002	N0003	N0004	N0005	N0006	N0007	N0008	...	N4166	N4167	N4168	N4169	N4170	N4171	N4
8718	2013-12-30 06:00:00	0.407	0.549	0.039	0.230	0.116	0.314	0.030	0.156	0.143	...	3.068	1.470	0.190	NaN	0.132	0.039	0.2
8719	2013-12-30 07:00:00	0.333	0.532	0.089	0.307	0.158	0.934	0.030	0.155	0.235	...	2.779	0.366	0.181	NaN	0.101	0.119	0.2
8720	2013-12-30 08:00:00	0.307	0.545	0.731	0.556	0.122	0.866	0.073	0.168	0.174	...	1.453	0.317	0.147	NaN	0.095	0.133	0.2
8721	2013-12-30 09:00:00	0.594	0.657	0.473	1.216	0.526	0.231	0.347	0.163	0.692	...	0.625	0.730	0.349	NaN	0.155	0.131	0.2
8722	2013-12-30 10:00:00	1.873	0.667	0.892	0.740	0.487	0.152	0.217	0.154	0.747	...	0.124	0.501	0.153	NaN	0.093	0.069	0.4
8723	2013-12-30 11:00:00	0.659	0.601	0.240	0.614	0.705	0.152	0.198	0.442	0.574	...	0.051	0.738	0.090	NaN	0.331	0.023	0.3
8724	2013-12-30 12:00:00	0.393	0.448	1.292	0.317	1.163	0.119	0.154	0.215	0.582	...	1.648	1.243	0.107	NaN	0.284	0.089	0.4
8725	2013-12-30 13:00:00	0.395	0.501	0.303	0.447	0.587	0.205	0.121	0.189	1.072	...	0.594	2.732	0.084	NaN	1.157	0.119	0.3
8726	2013-12-30 14:00:00	0.440	0.540	0.341	0.572	0.573	0.222	0.137	0.191	0.365	...	0.043	2.171	0.169	NaN	0.736	0.117	0.3
8727	2013-12-30 15:00:00	0.637	0.517	0.171	0.553	1.109	0.229	0.079	0.203	0.874	...	0.394	2.118	0.069	NaN	0.315	0.114	0.3
8728	2013-12-30 16:00:00	0.309	0.547	0.555	0.614	0.915	0.243	0.034	0.302	1.148	...	0.344	0.778	0.100	NaN	0.523	0.081	0.4
8729	2013-12-30 17:00:00	0.704	0.515	0.242	1.042	2.535	0.241	0.031	0.463	0.752	...	2.888	0.392	0.151	NaN	0.314	2.351	1.5
8730	2013-12-30 18:00:00	0.280	0.567	0.253	0.874	3.414	0.434	0.029	0.795	0.228	...	3.164	1.963	0.185	NaN	0.490	0.440	0.7
8731	2013-12-30 19:00:00	1.625	0.503	0.459	0.802	1.259	0.322	0.033	0.341	0.237	...	2.618	2.274	0.212	NaN	0.329	0.142	0.3
8732	2013-12-30 20:00:00	0.348	0.567	0.323	1.810	1.513	0.931	0.157	0.287	0.297	...	2.215	1.745	0.200	NaN	0.515	0.115	0.3
8733	2013-12-30 21:00:00	0.482	0.500	0.131	1.322	1.530	0.559	0.101	0.344	0.199	...	1.592	2.724	0.437	NaN	0.282	0.115	0.2
8734	2013-12-30 22:00:00	0.555	0.787	0.073	1.713	1.367	0.550	0.112	0.326	0.194	...	0.097	1.547	0.245	NaN	0.213	0.261	0.3
8735	2013-12-30 23:00:00	0.508	0.608	0.107	0.241	0.956	0.381	0.103	0.634	0.250	...	0.038	1.199	0.246	NaN	0.285	0.110	0.2

3432 rows × 4177 columns

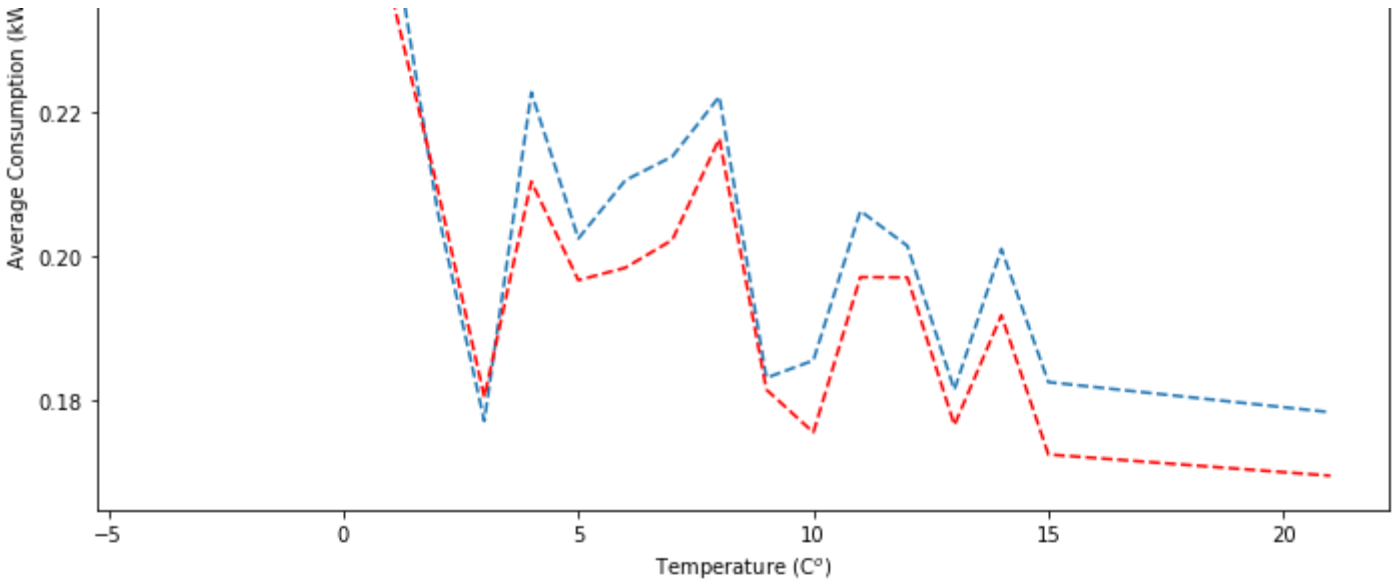
In [105]:
```
# TOU low price vs estimation of TOU group facing default price using non-TOU data
# hourly energy consumption v.s. temperature in event days
fig_all = plt.figure(figsize = (10,150))
ax_Ntou = [] # store subplot objects
ax_tou = [] # store subplot objects
for i in range(24):
    ax_Ntou.append(fig_all.add_subplot(24, 1, i+1))
    if i <= 9:
        # calculate the average difference between two
        # calculate the average consumption grouped by different temperatures
        df_dVSt = df_tou1h_lPrice[df_tou1h_lPrice.GMT.str.contains('0' + str(i) + ':00:00')] #TOU demand vs t
emperature for high price periods
        df_dVSt = df_dVSt.groupby('TempC')['Total'].mean() / (df_dVSt.shape[1] - 4)
        df_dVSt = df_dVSt.reset_index()
        ax_Ntou[-1].plot(df_dVSt.TempC, df_dVSt.Total,  c = 'C0', label = 'TOU Low Price', linestyle='dashed'
)

        df_NdVSt_help = df_Ntou1h[df_tariff_1h.Price == 0.0399]
        df_NdVSt = df_NdVSt_help[df_NdVSt_help.GMT.str.contains('0' + str(i) + ':00:00')] # estimation of TOU
group facing default price using non-TOU data (will remove the const difference later)
        df_NdVSt = df_NdVSt.groupby('TempC')['Total'].mean() / (df_NdVSt.shape[1] - 4)
        df_NdVSt = df_NdVSt.reset_index()
        ax_Ntou[-1].plot(df_NdVSt.TempC, df_NdVSt.Total - const_shift[i], c = 'red', label = 'non-TOU Estimat
ion', linestyle='dashed' )

        ax_Ntou[-1].set_title('Time: 0' + str(i) + ':00')
        ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_Ntou[-1].set_ylabel('Average Consumption (kWh)')
        ax_Ntou[-1].legend()
    else:
        df_dVSt = df_tou1h_lPrice[df_tou1h_lPrice.GMT.str.contains(str(i) + ':00:00')] #TOU demand vs tempera
ture for high price periods
        df_dVSt = df_dVSt.groupby('TempC')['Total'].mean() / (df_dVSt.shape[1] - 4)
        df_dVSt = df_dVSt.reset_index()
        ax_Ntou[-1].plot(df_dVSt.TempC, df_dVSt.Total,  c = 'C0', label = 'TOU Low Price', linestyle='dashed'
)

        df_NdVSt_help = df_Ntou1h[df_tariff_1h.Price == 0.0399]
        df_NdVSt = df_NdVSt_help[df_NdVSt_help.GMT.str.contains(str(i) + ':00:00')] # estimation of TOU group
facing default price using non-TOU data (will remove the const difference later)
        df_NdVSt = df_NdVSt.groupby('TempC')['Total'].mean() / (df_NdVSt.shape[1] - 4)
        df_NdVSt = df_NdVSt.reset_index()
        ax_Ntou[-1].plot(df_NdVSt.TempC, df_NdVSt.Total - const_shift[i], c = 'red', label = 'non-TOU Estimat
ion', linestyle='dashed' )

        ax_Ntou[-1].set_title('Time: ' + str(i) + ':00')
        ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_Ntou[-1].set_ylabel('Average Consumption (kWh)')
        ax_Ntou[-1].legend()
plt.tight_layout()
```

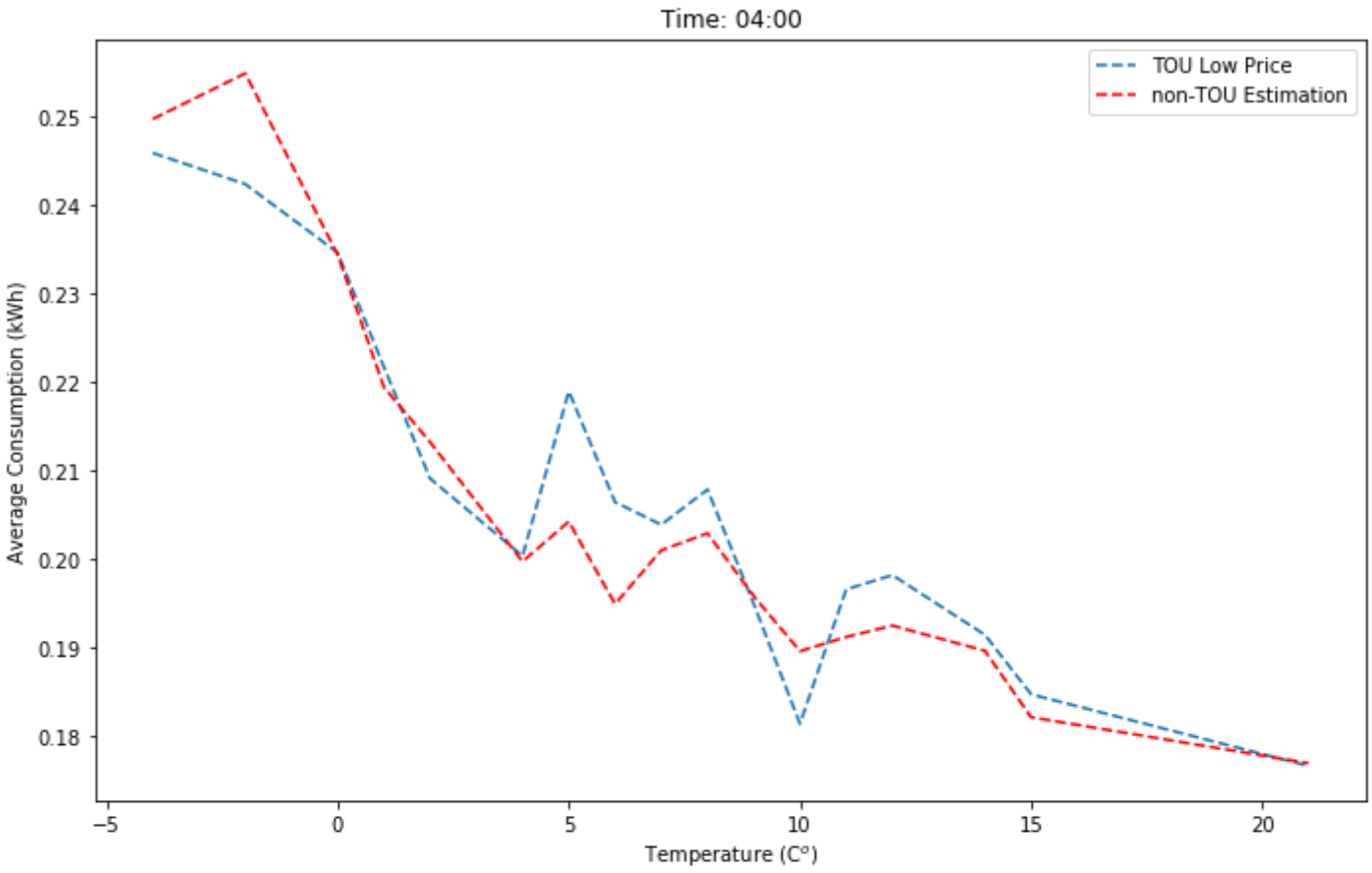

Time: 00:00
TOU Low Price
non-TOU Estimation
Average Consumption (kWh)
0.40
0.35
0.30
0.25
0.20
−5
0
5
10
15
20
Temperature (C°)

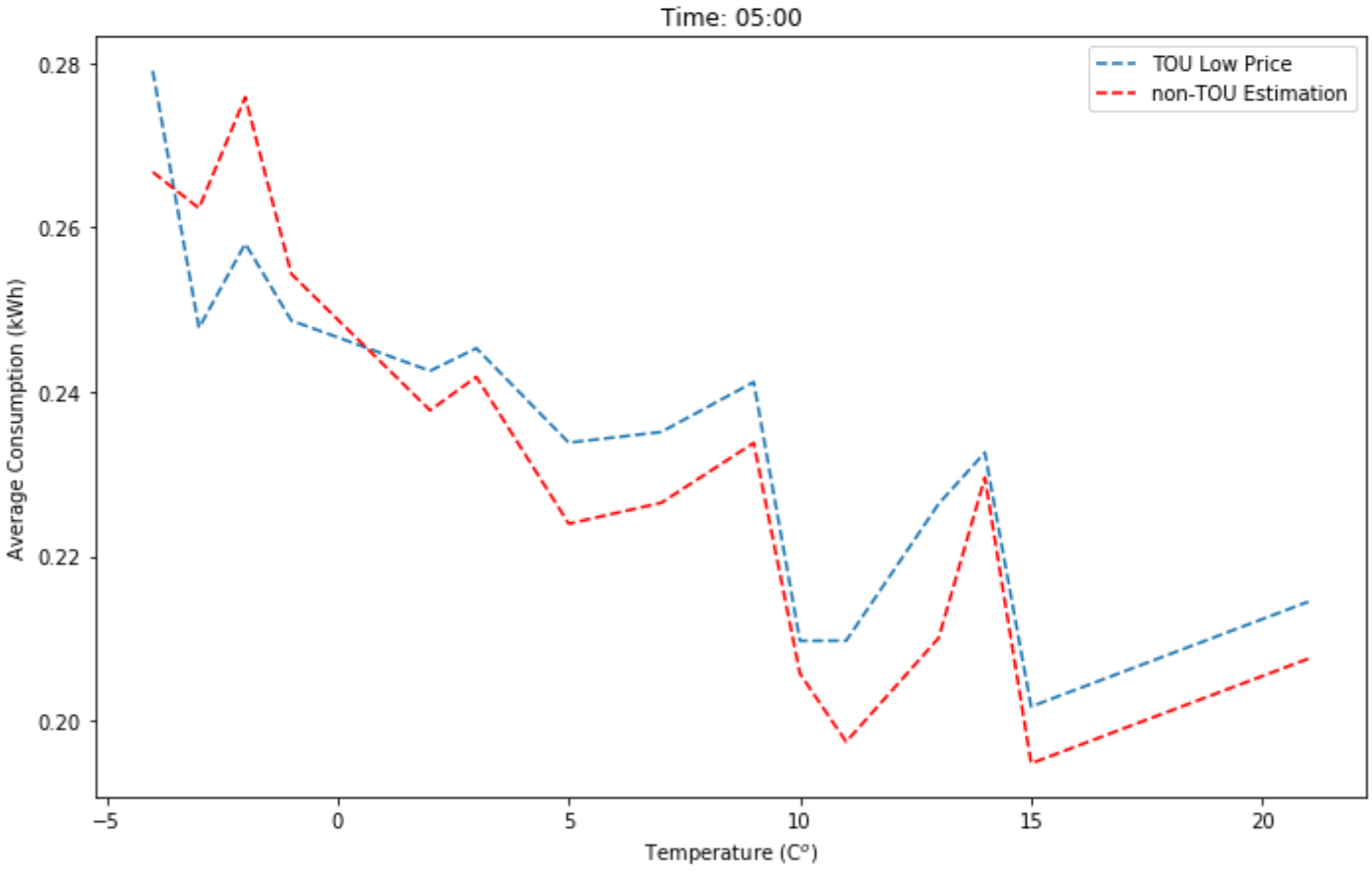

Time: 01:00
TOU Low Price
non-TOU Estimation
Average Consumption (kWh)
0.34
0.32
0.30
0.28
0.26
0.24
0.22
0.20
0.18
−5
0
5
10
15
20
Temperature (C°)

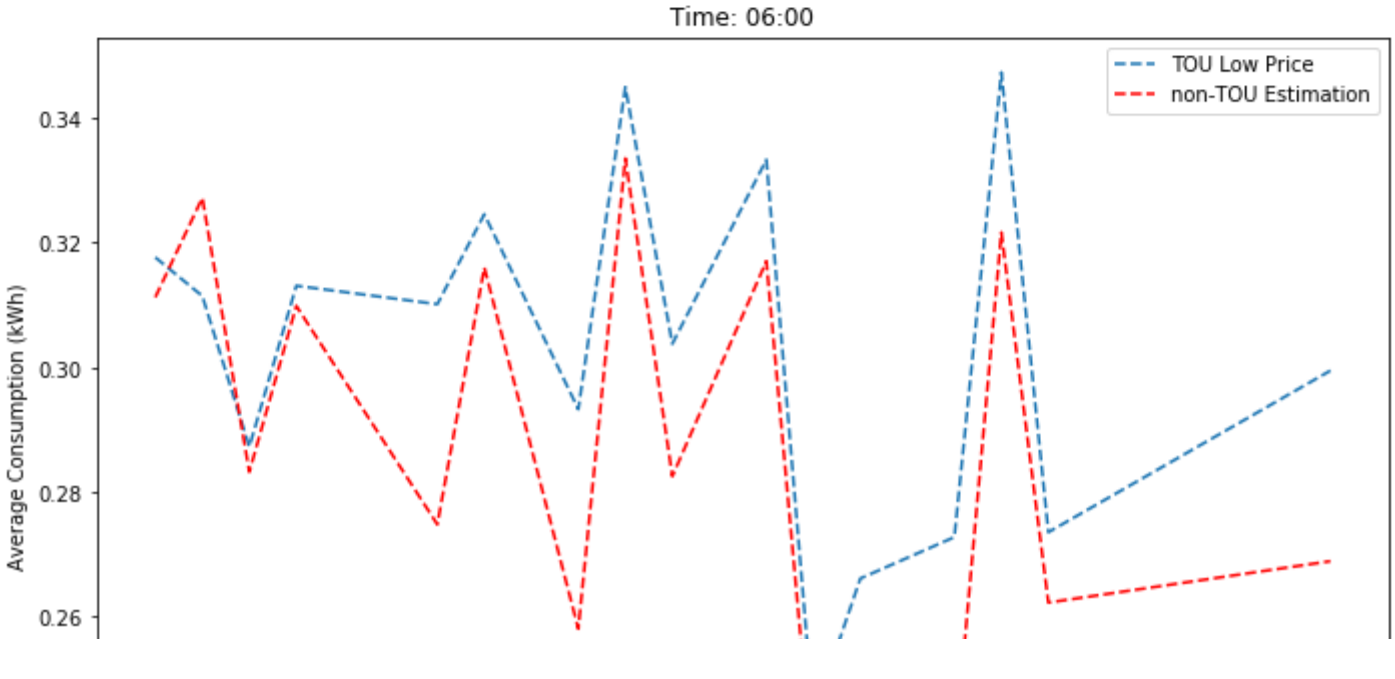

Time: 02:00
TOU Low Price
non-TOU Estimation
Average Consumption (kWh)
0.28
0.26
0.24
0.22
0.20
0.18
−5
0
5
10
15
20
Temperature (C°)

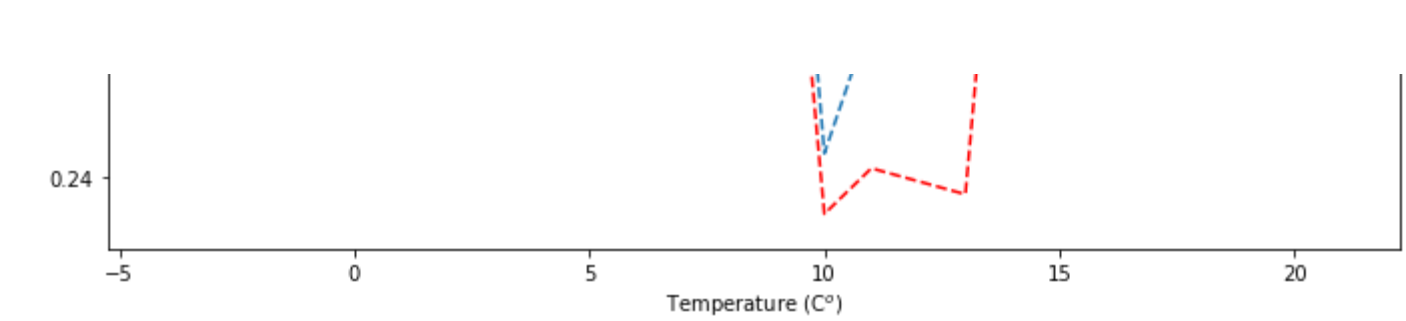

Time: 03:00
TOU Low Price
non-TOU Estimation
0.26
0.24

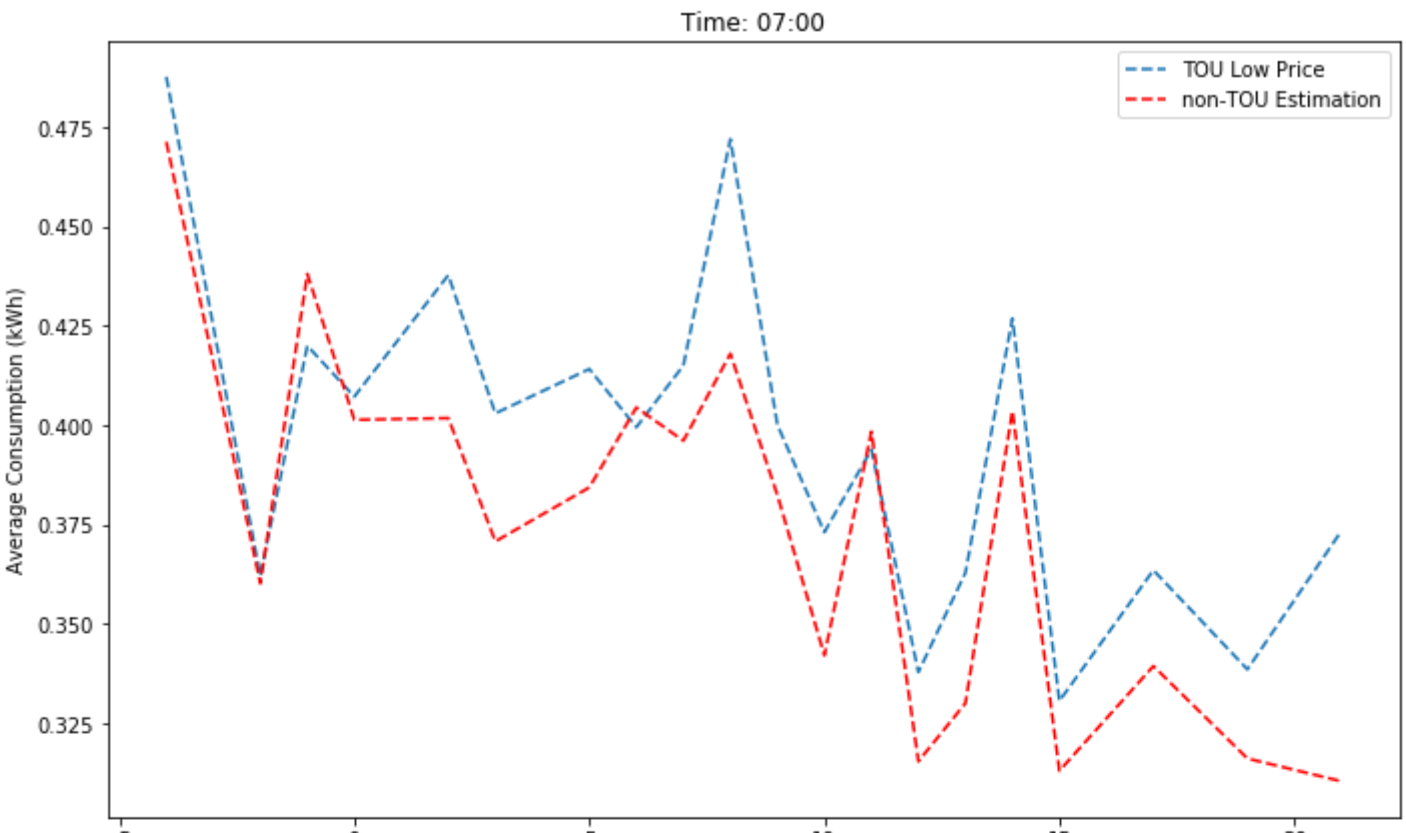

Average Consumption (kW
0.22
0.20
0.18
−5
0
5
10
15
20
Temperature (C°)

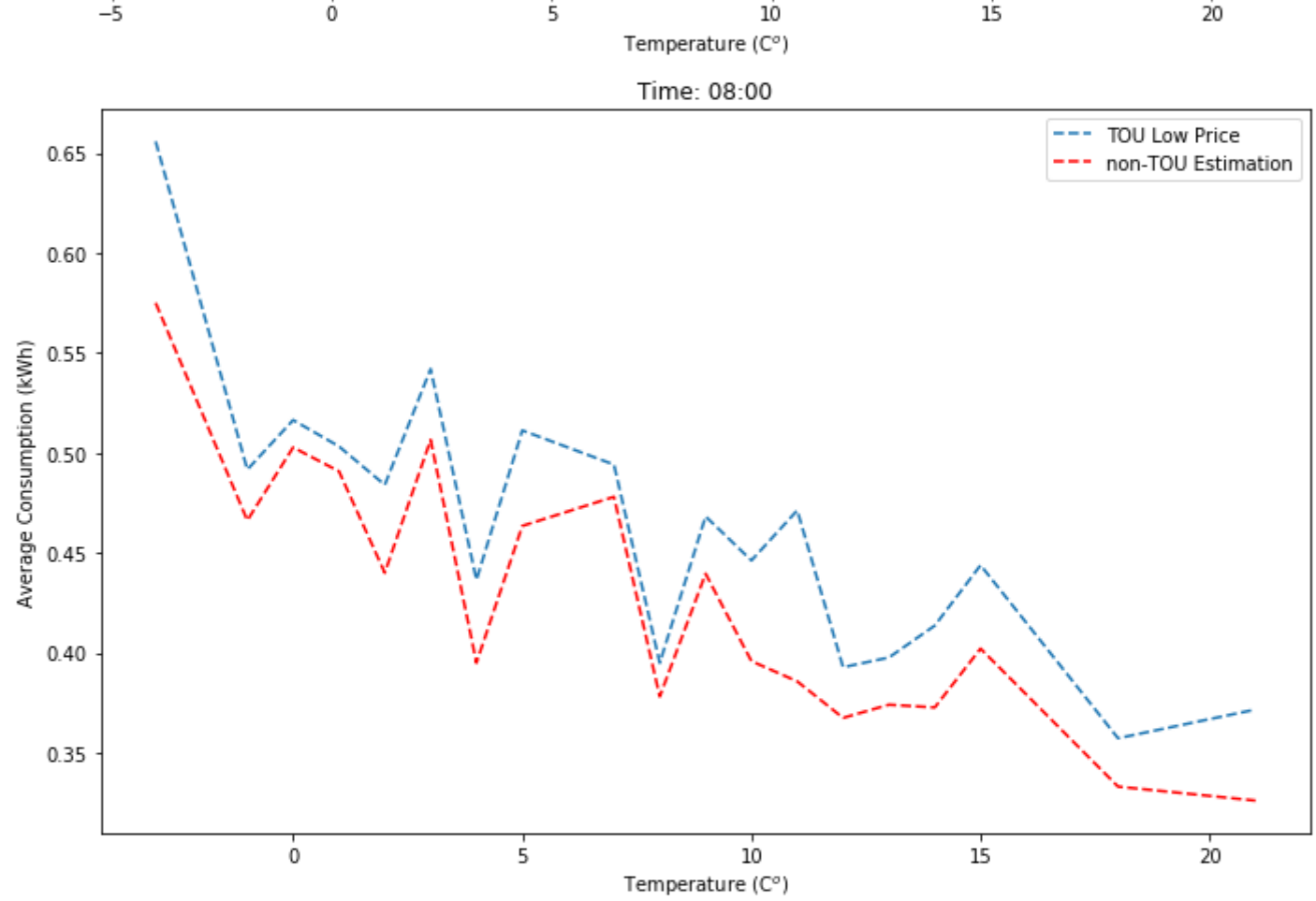

Time: 04:00
TOU Low Price
non-TOU Estimation
Average Consumption (kWh)
0.25
0.24
0.23
0.22
0.21
0.20
0.19
0.18
−5
0
5
10
15
20
Temperature (C°)

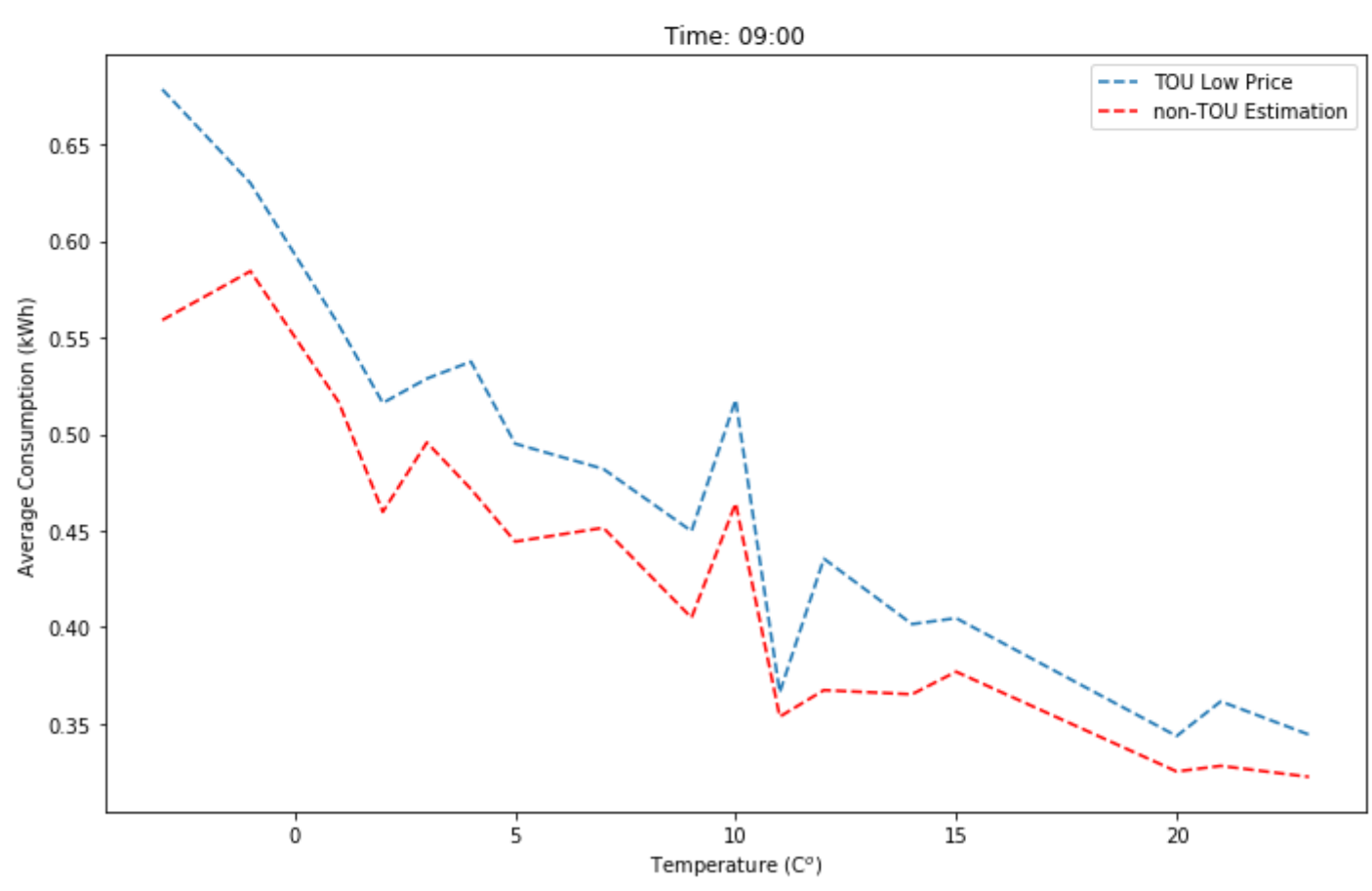

Time: 05:00
TOU Low Price
non-TOU Estimation
Average Consumption (kWh)
0.28
0.26
0.24
0.22
0.20
−5
0
5
10
15
20
Temperature (C°)

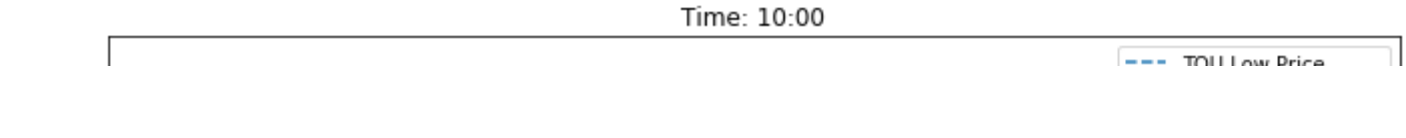

Time: 06:00
TOU Low Price
non-TOU Estimation
Average Consumption (kWh)
0.34
0.32
0.30
0.28
0.26

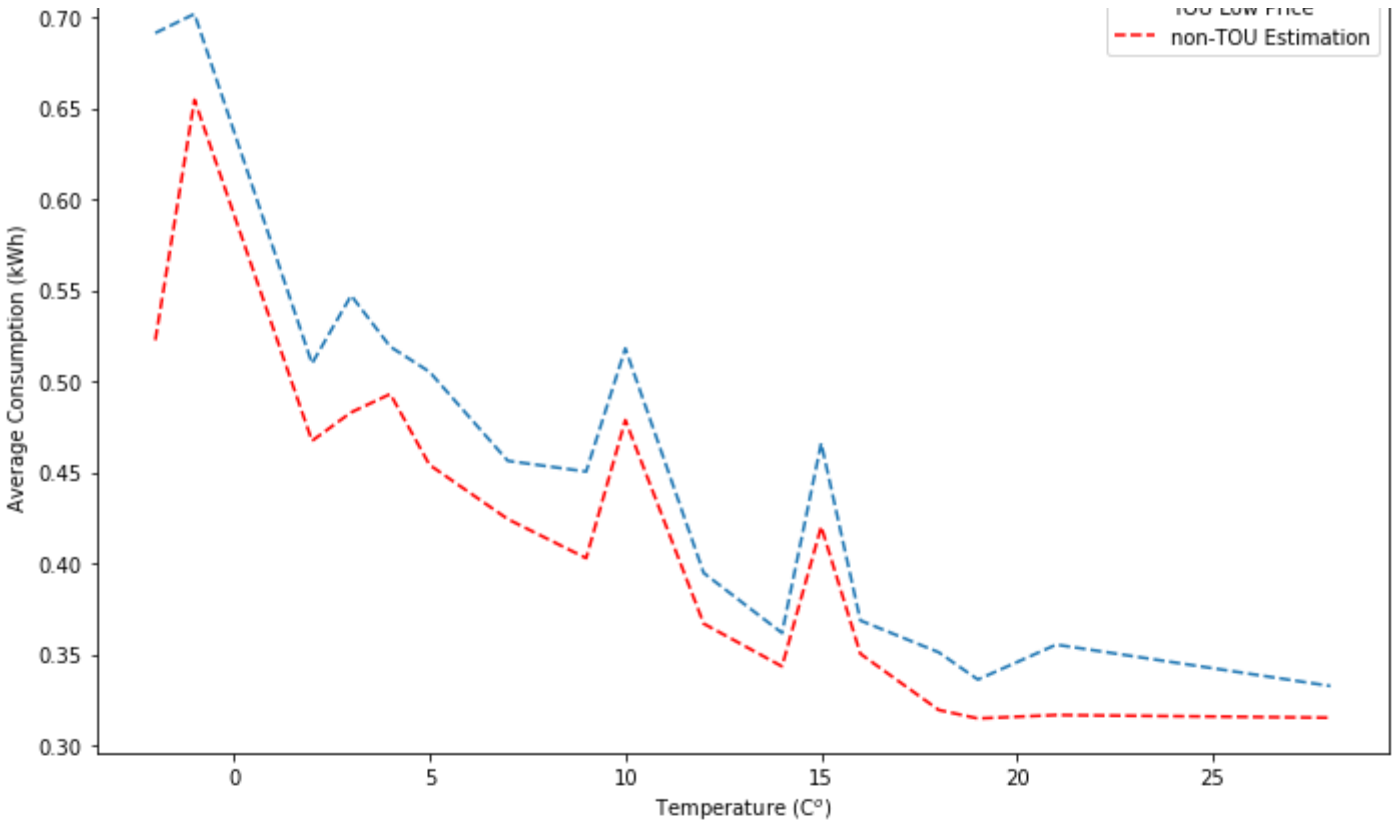

0.24
−5
0
5
10
15
20
Temperature (C°)


Time: 07:00
TOU Low Price
non-TOU Estimation
Average Consumption (kWh)
0.475
0.450
0.425
0.400
0.375
0.350
0.325
0
5
10
15
20
Temperature (C°)

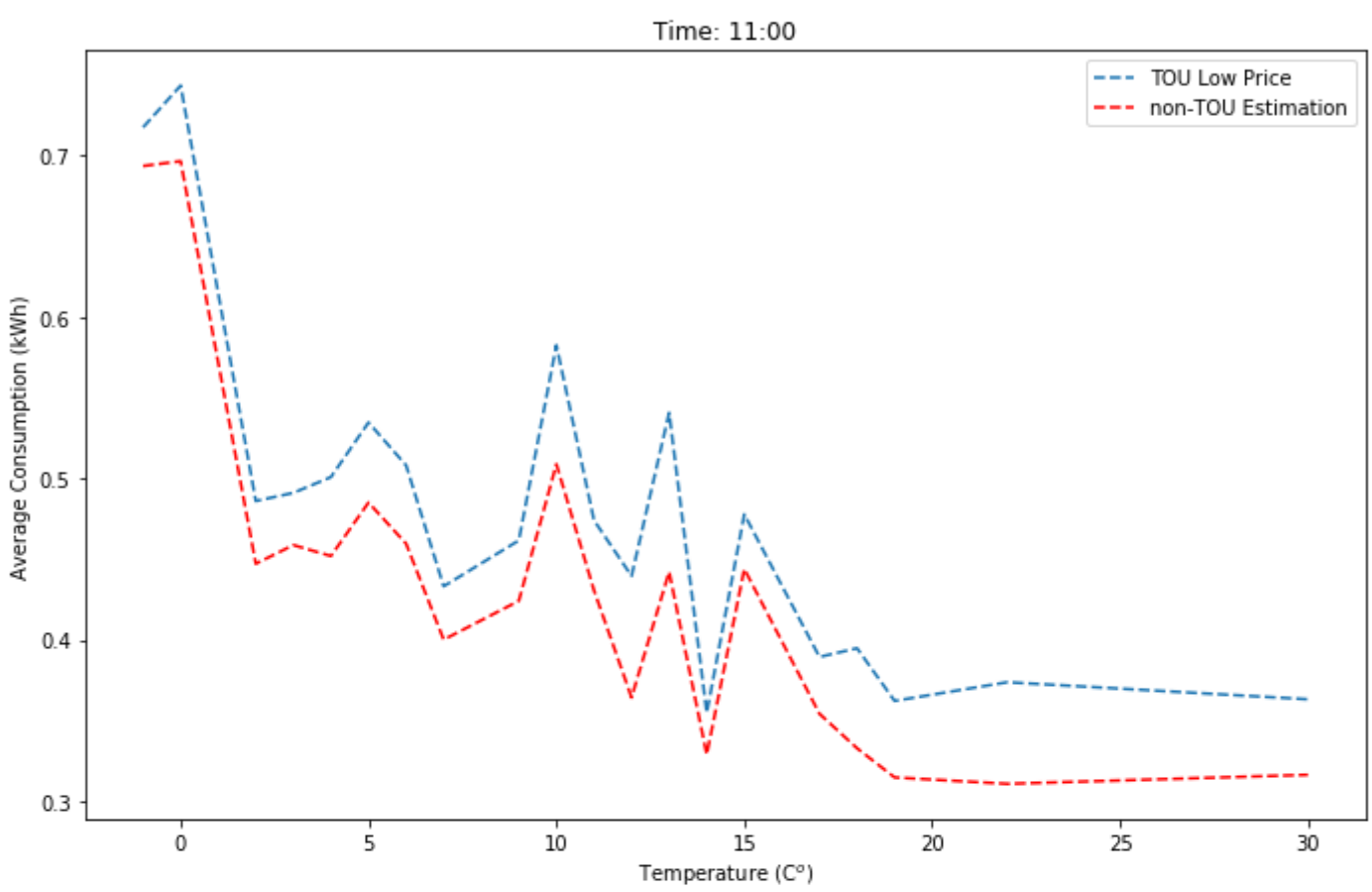

Time: 08:00
TOU Low Price
non-TOU Estimation
Average Consumption (kWh)
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0
5
10
15
20
Temperature (C°)


Time: 09:00
TOU Low Price
non-TOU Estimation
Average Consumption (kWh)
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0
5
10
15
20
Temperature (C°)

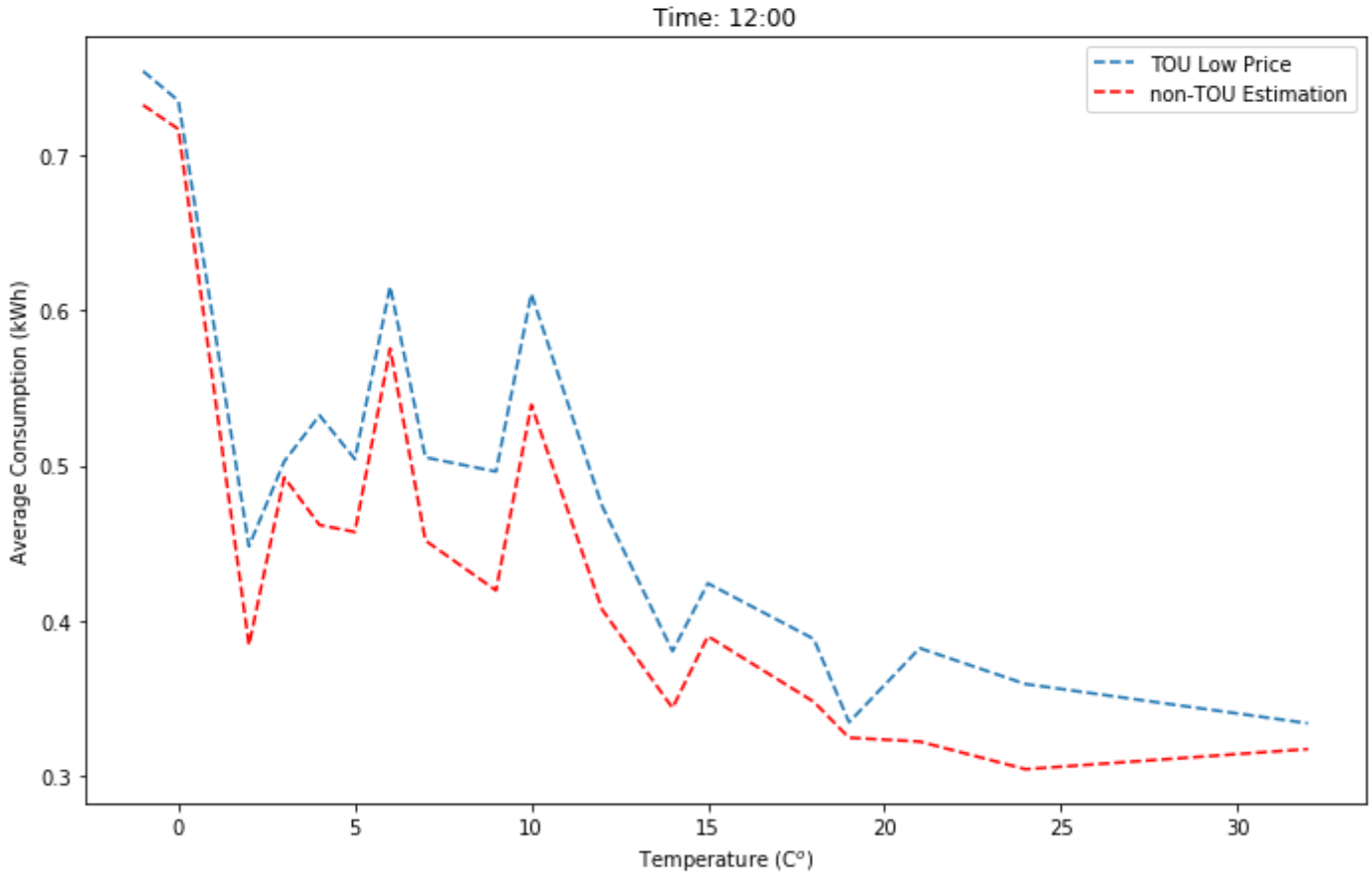

Time: 10:00



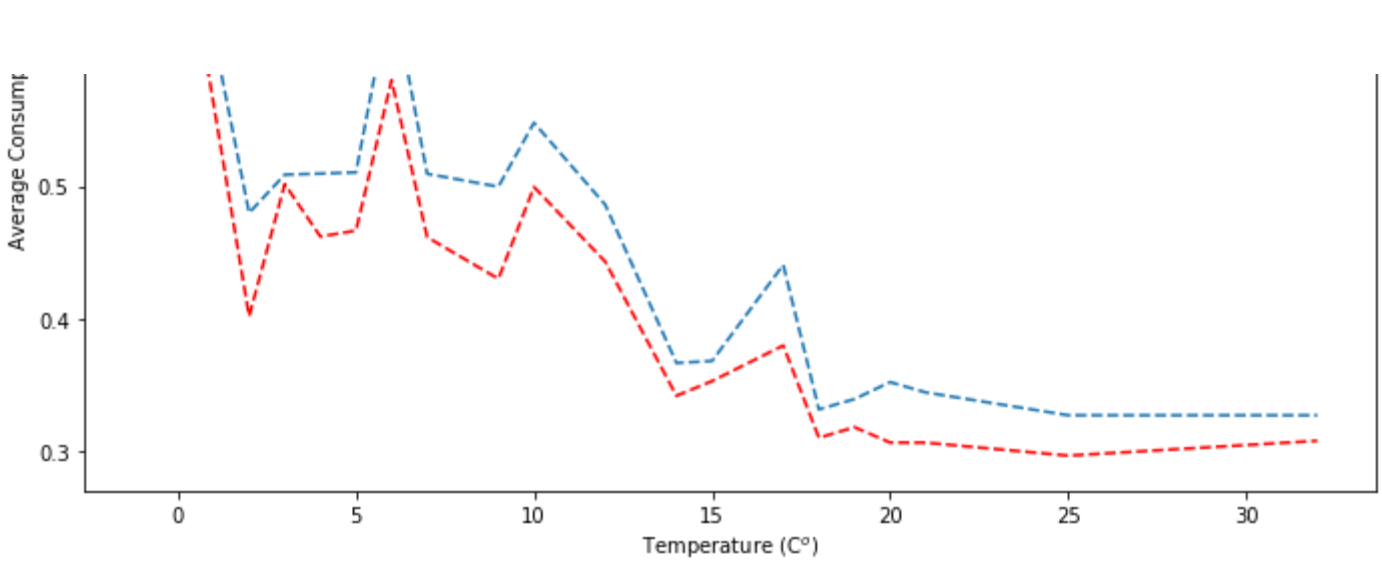
Time: 11:00

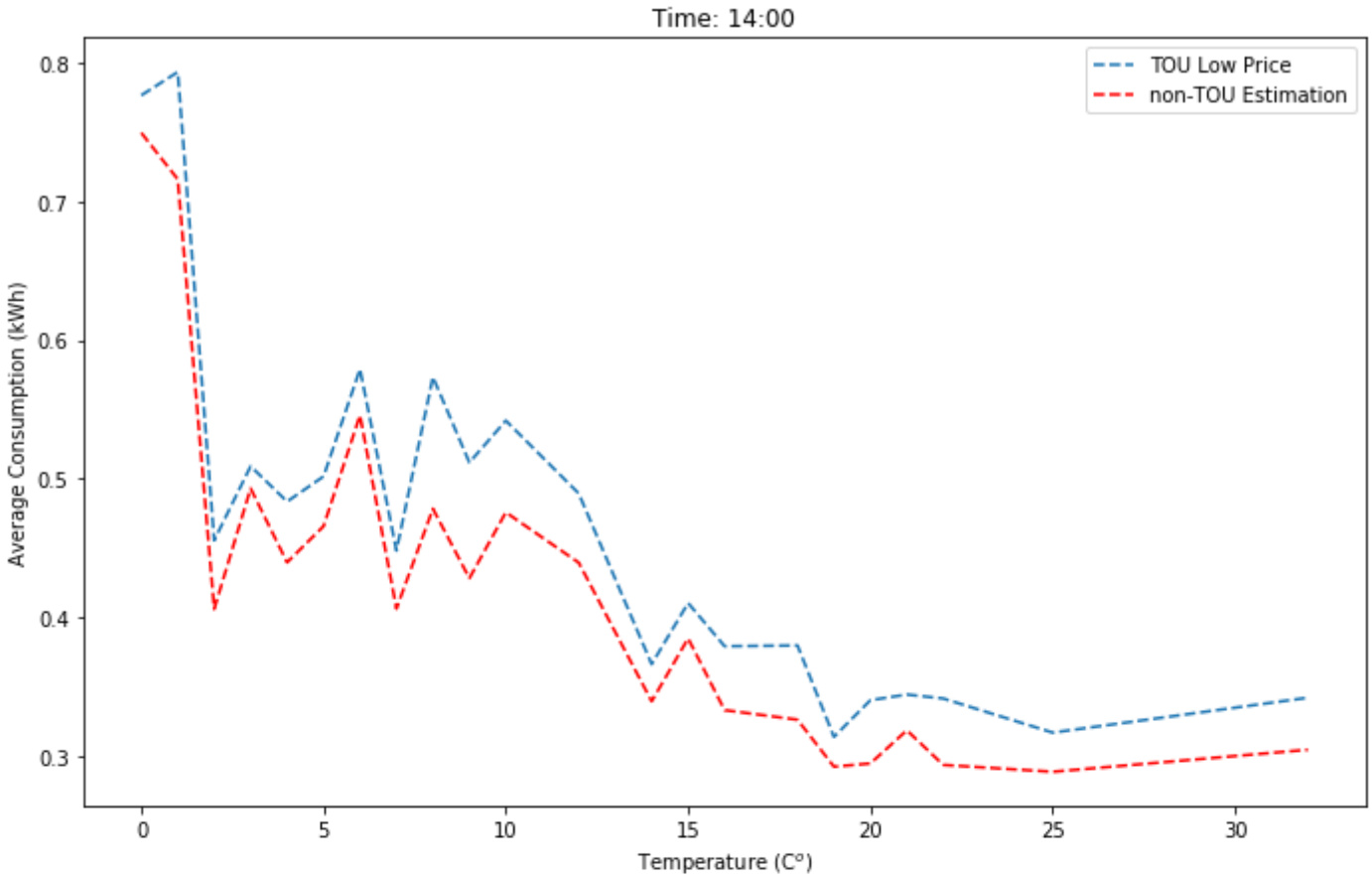


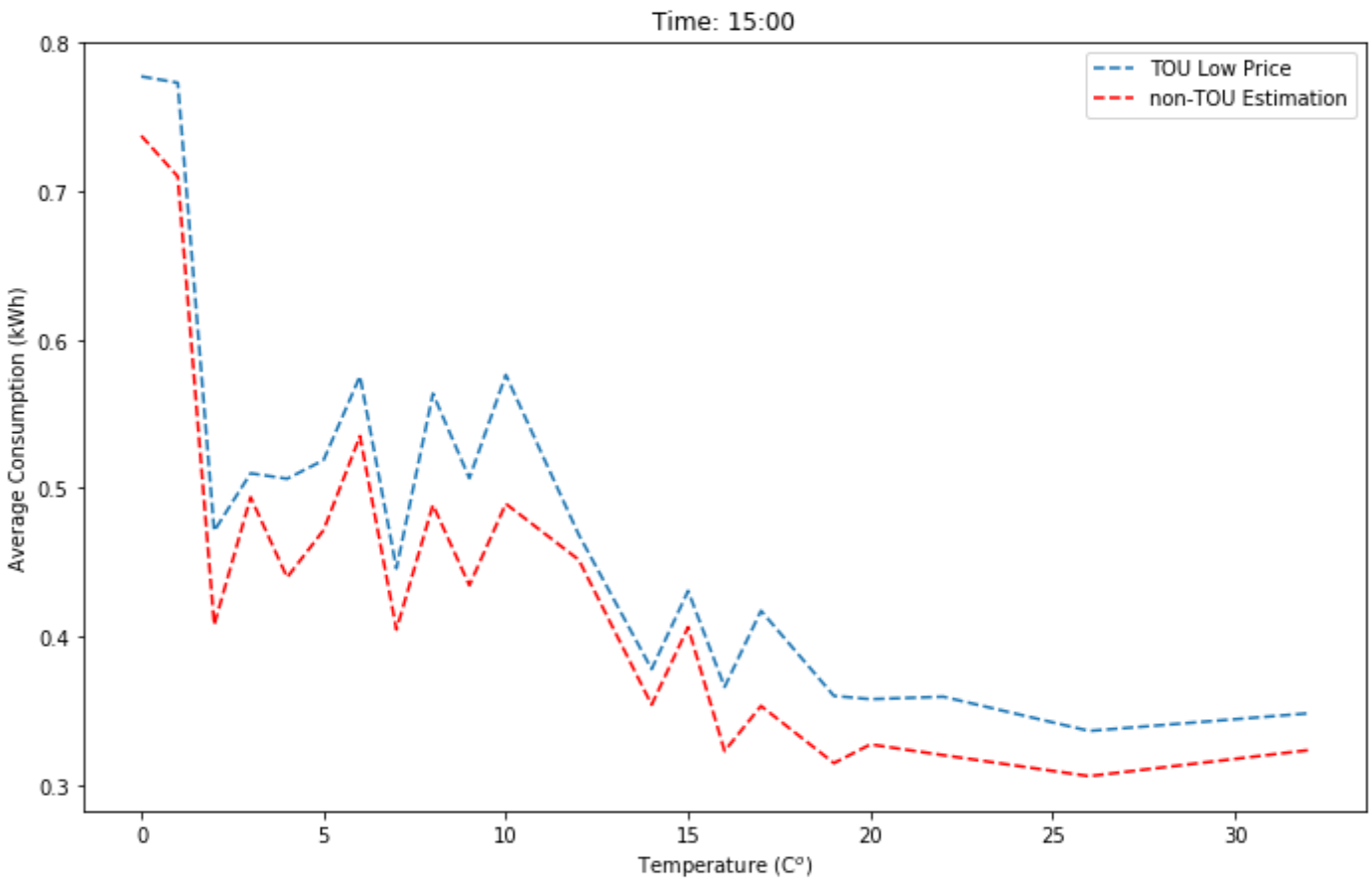
Time: 12:00

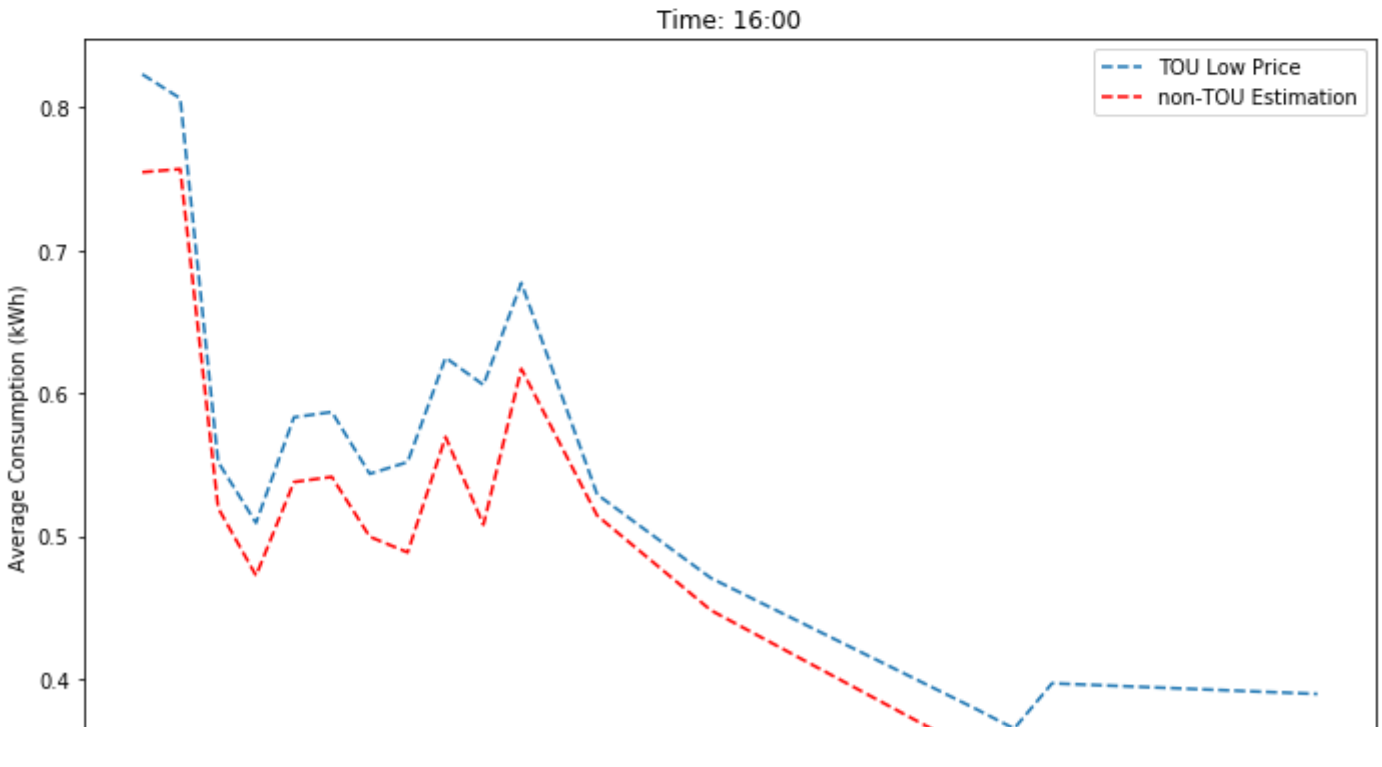


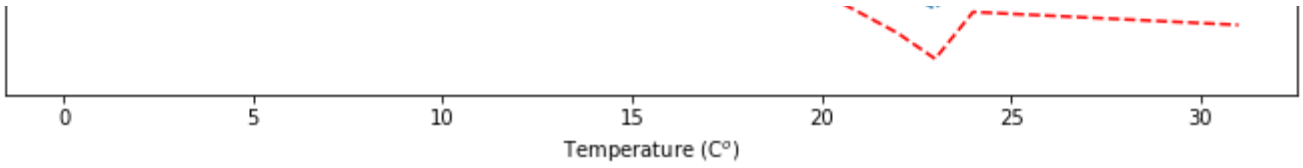
Time: 13:00

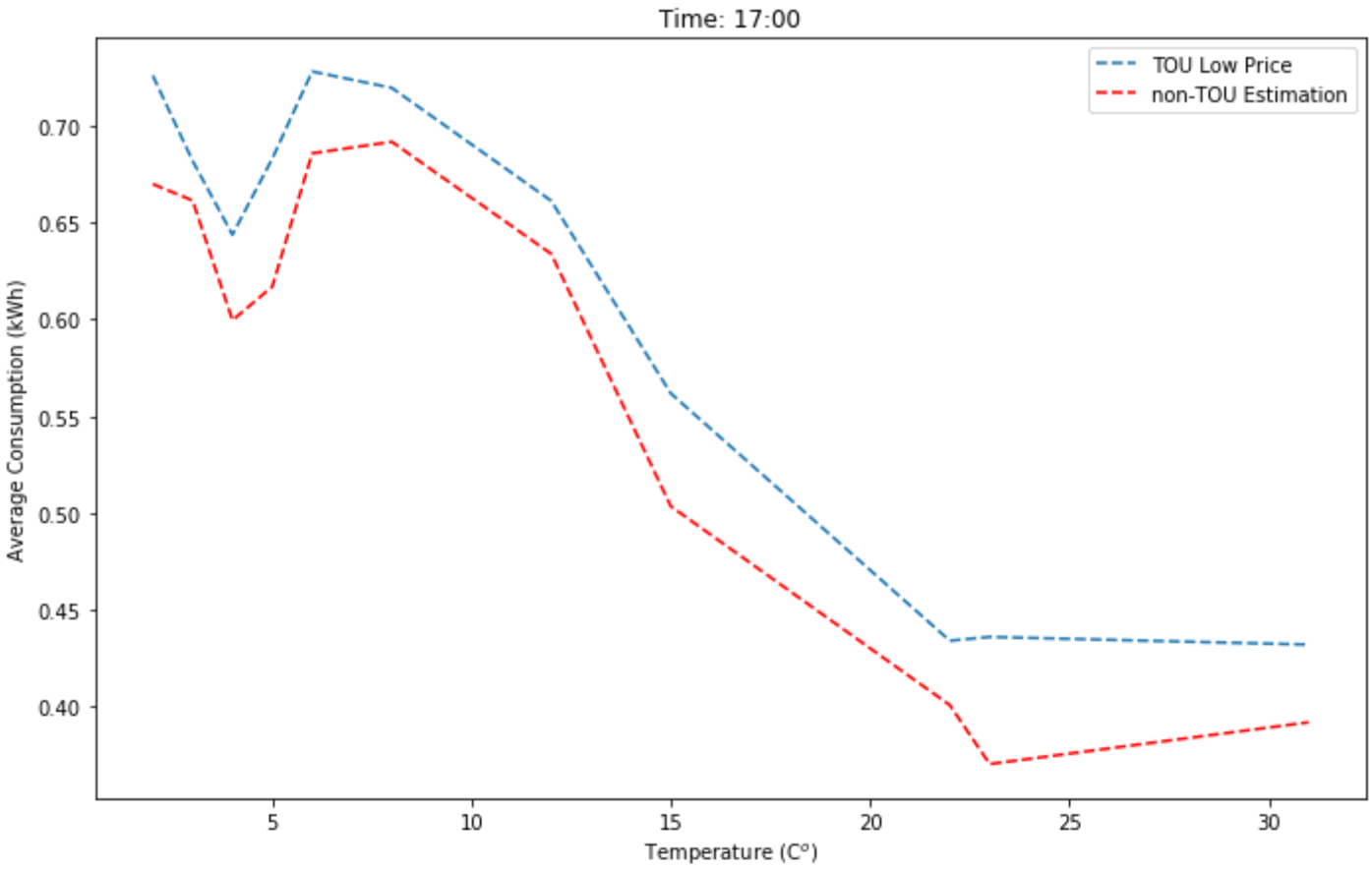

Average Consump
0.5
0.4
0.3
0
5
10
15
20
25
30
Temperature (C°)

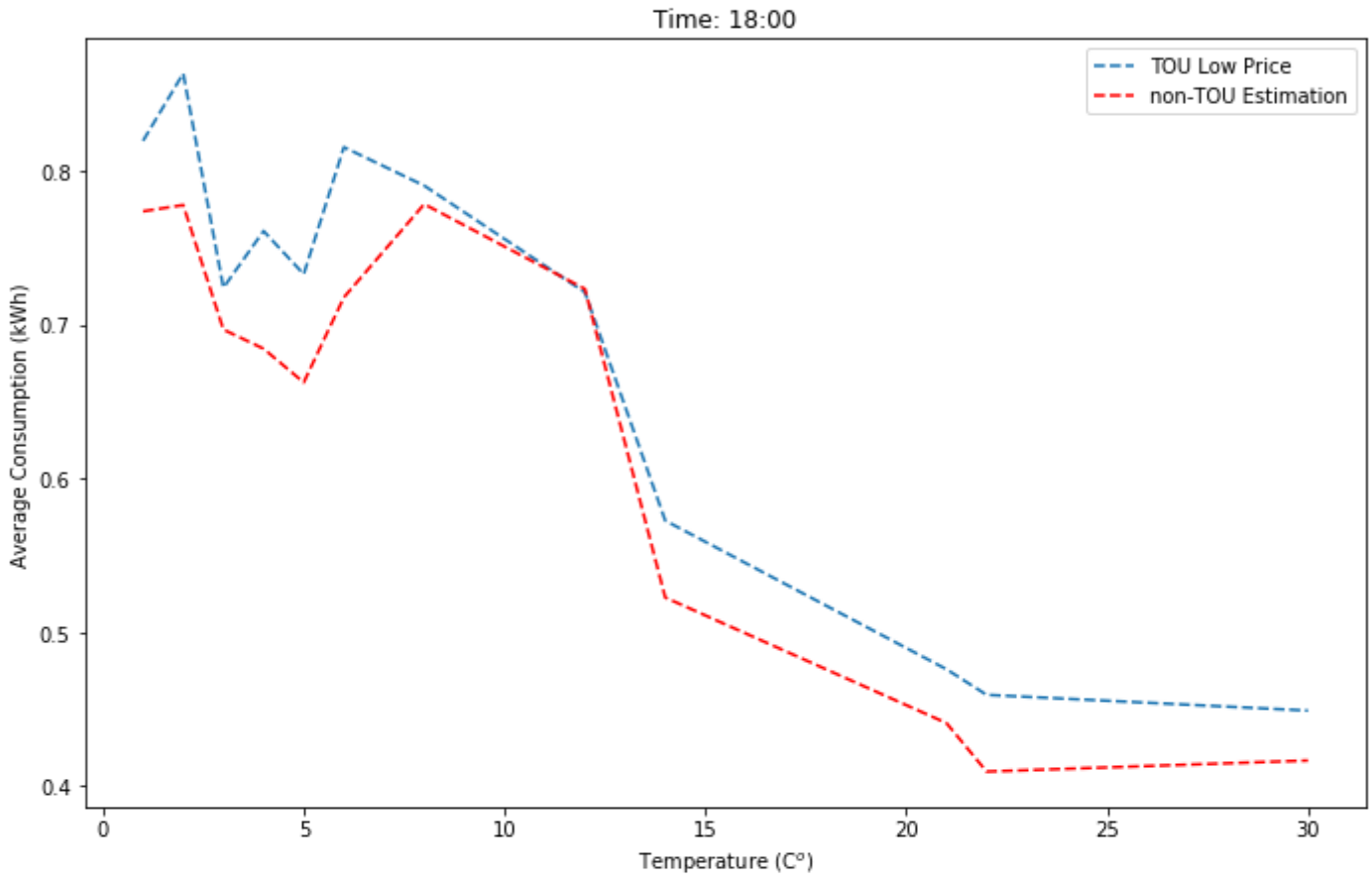

Time: 14:00
TOU Low Price
non-TOU Estimation
Average Consumption (kWh)
0.8
0.7
0.6
0.5
0.4
0.3
0
5
10
15
20
25
30
Temperature (C°)

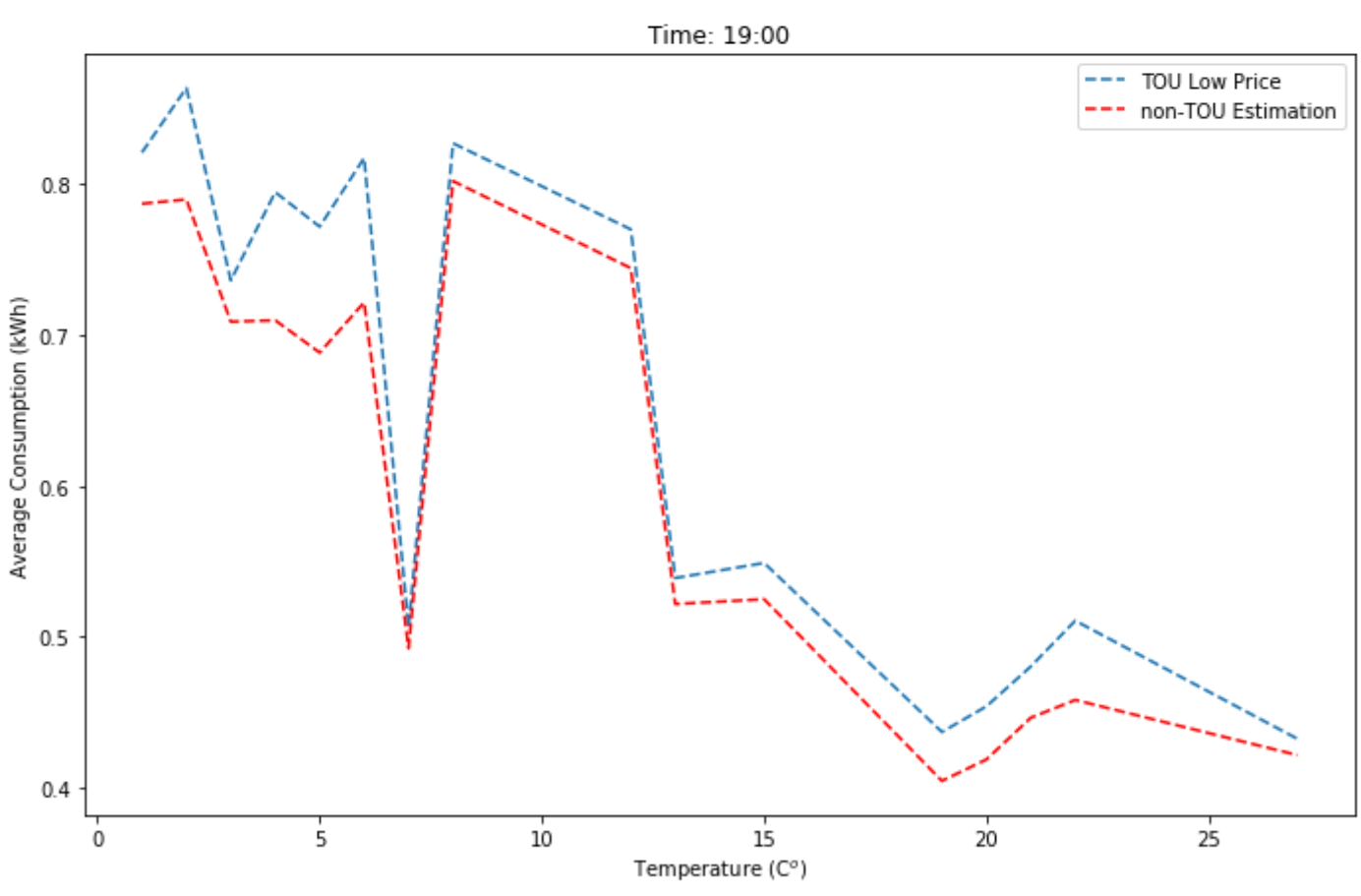

Time: 15:00
TOU Low Price
non-TOU Estimation
Average Consumption (kWh)
0.8
0.7
0.6
0.5
0.4
0.3
0
5
10
15
20
25
30
Temperature (C°)

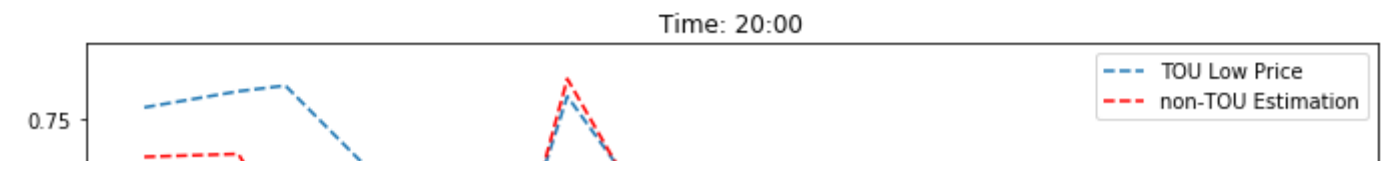

Time: 16:00
TOU Low Price
non-TOU Estimation
Average Consumption (kWh)
0.8
0.7
0.6
0.5
0.4

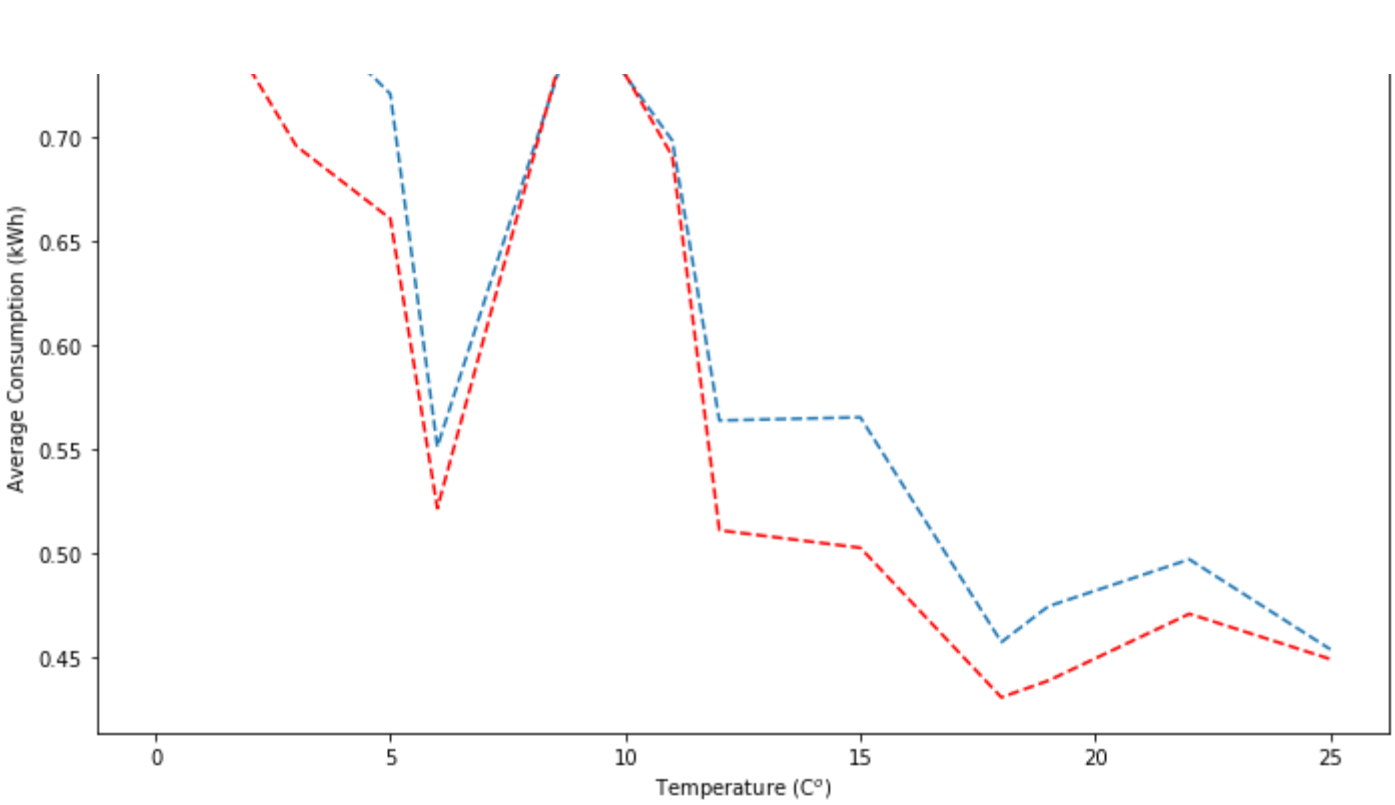

0
5
10
15
20
25
30
Temperature (C°)

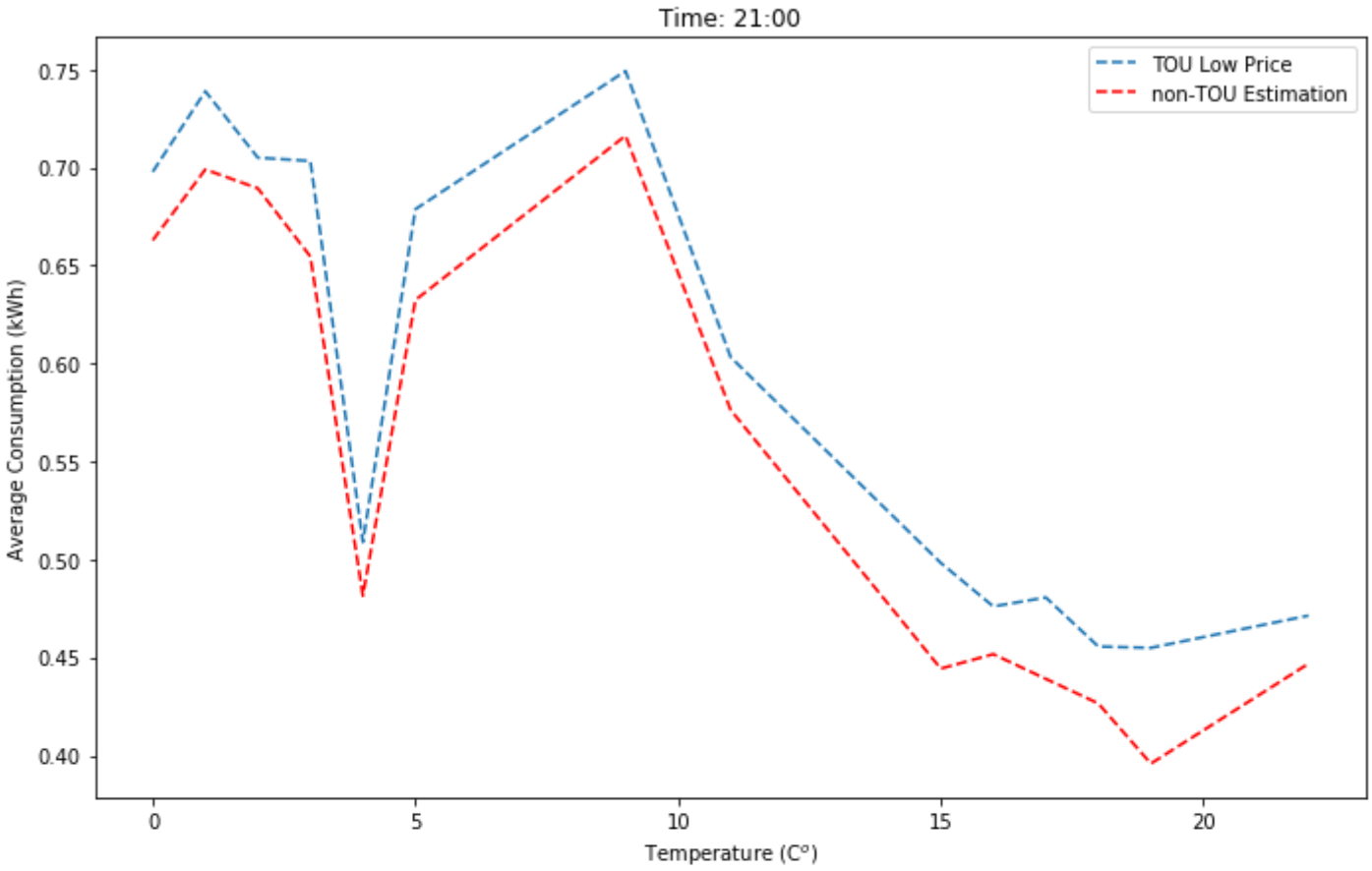

Time: 17:00
TOU Low Price
non-TOU Estimation
Average Consumption (kWh)
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0
5
10
15
20
25
30
Temperature (C°)

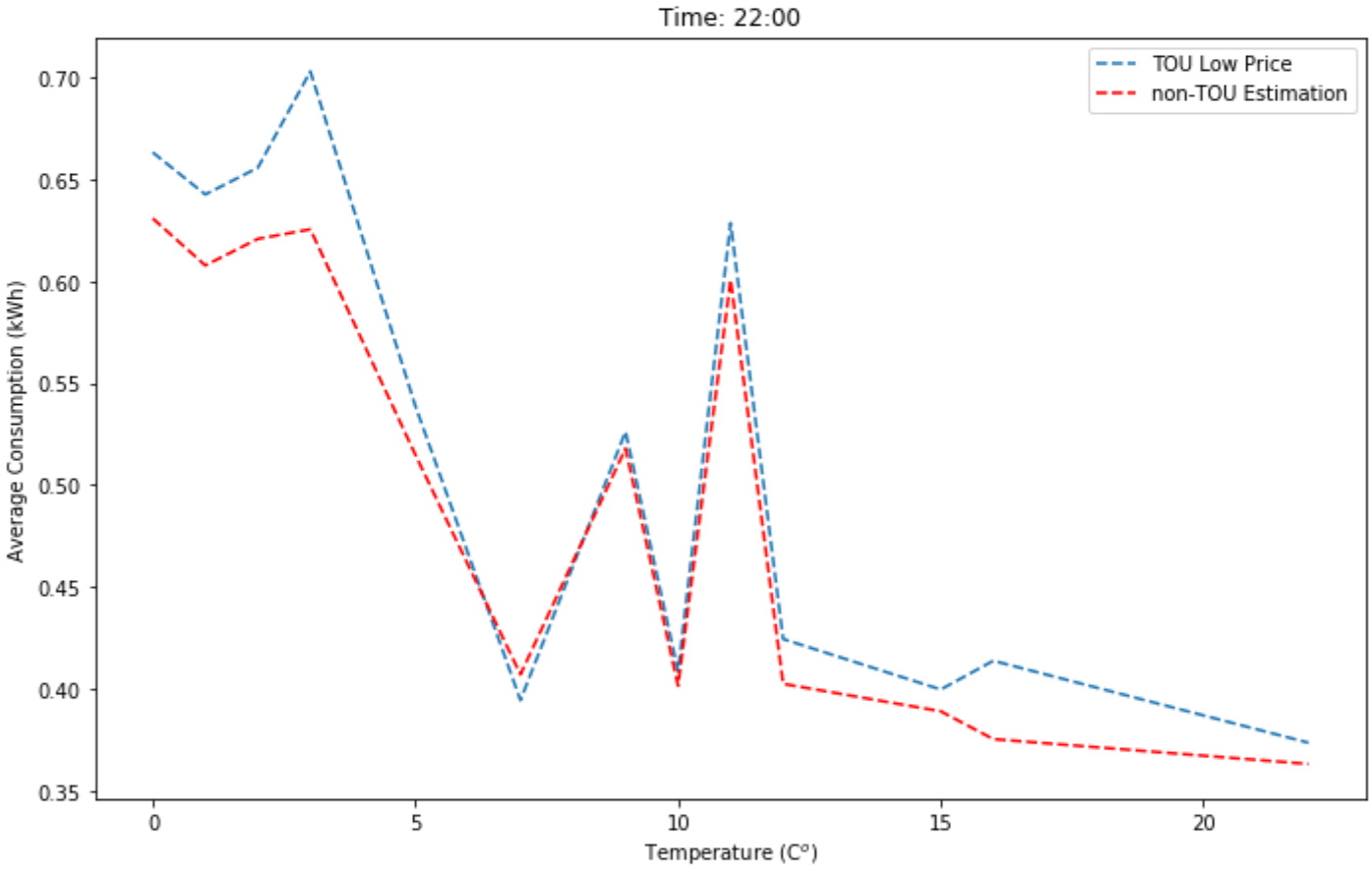

Time: 18:00
TOU Low Price
non-TOU Estimation
Average Consumption (kWh)
0.8
0.7
0.6
0.5
0.4
0
5
10
15
20
25
30
Temperature (C°)

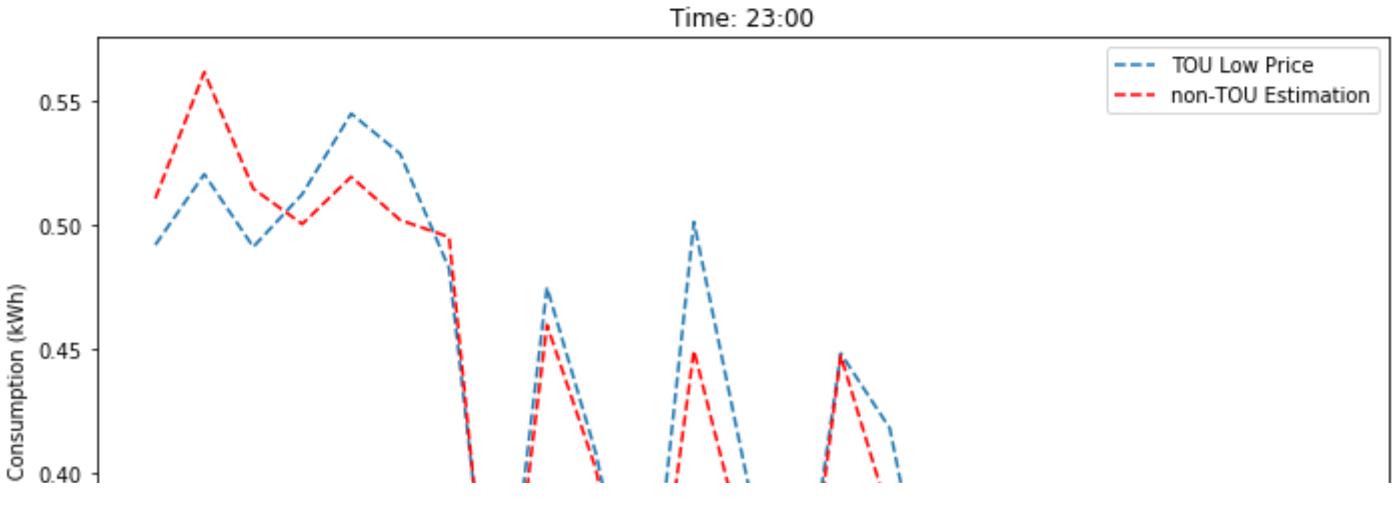

Time: 19:00
TOU Low Price
non-TOU Estimation
Average Consumption (kWh)
0.8
0.7
0.6
0.5
0.4
0
5
10
15
20
25
Temperature (C°)

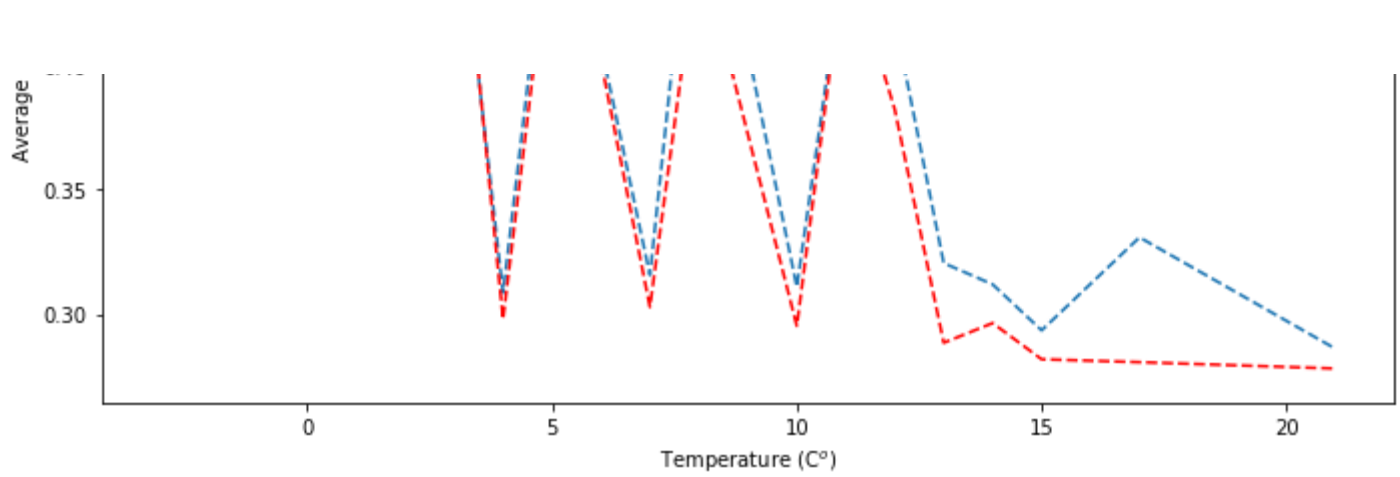

Time: 20:00
TOU Low Price
non-TOU Estimation
0.75


0.70
0.65
0.60
0.55
0.50
0.45
Average Consumption (kWh)
0
5
10
15
20
25
Temperature (C°)


Time: 21:00
TOU Low Price
non-TOU Estimation
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
Average Consumption (kWh)
0
5
10
15
20
Temperature (C°)


Time: 22:00
TOU Low Price
non-TOU Estimation
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
Average Consumption (kWh)
0
5
10
15
20
Temperature (C°)


Time: 23:00
TOU Low Price
non-TOU Estimation
0.55
0.50
0.45
0.40
Consumption (kWh)

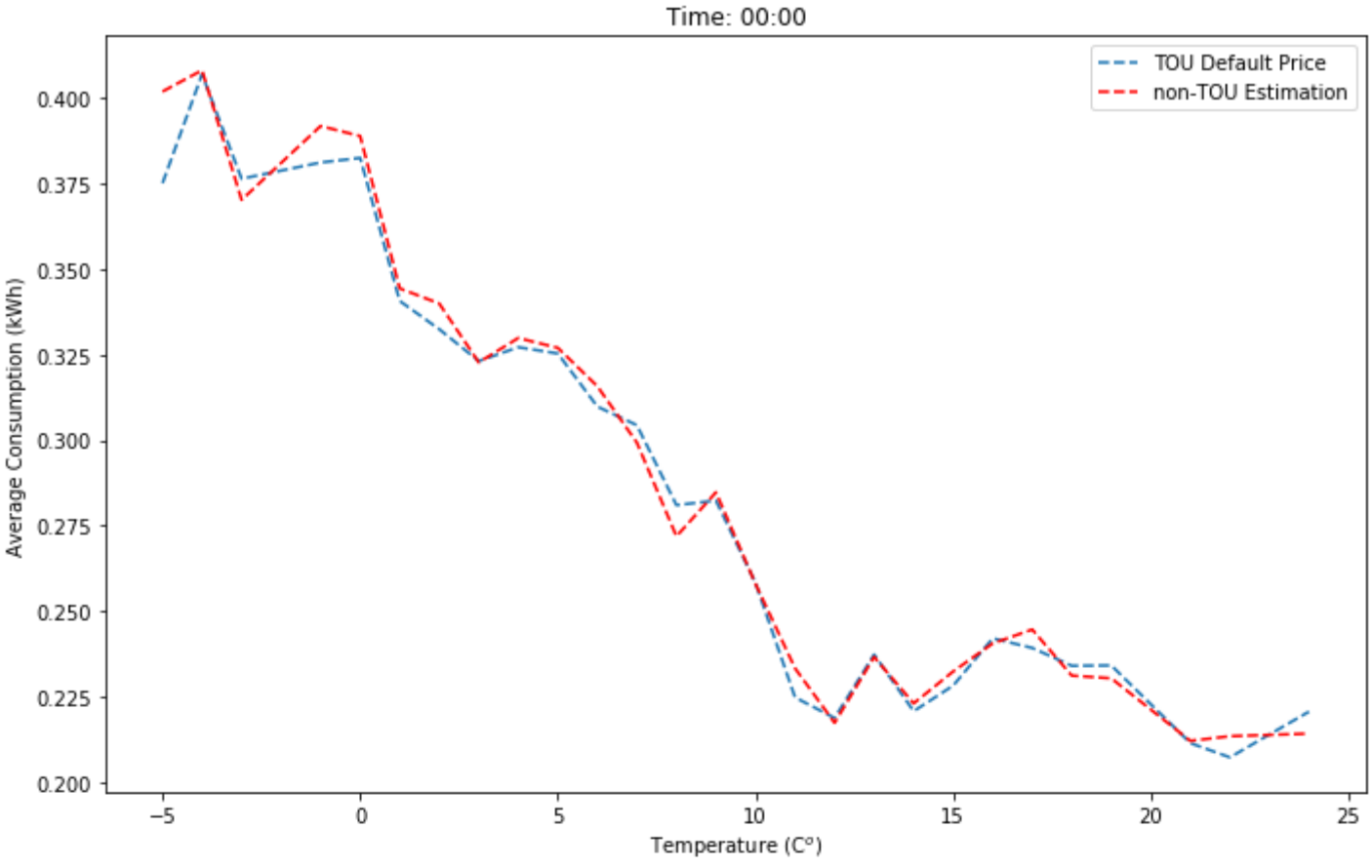


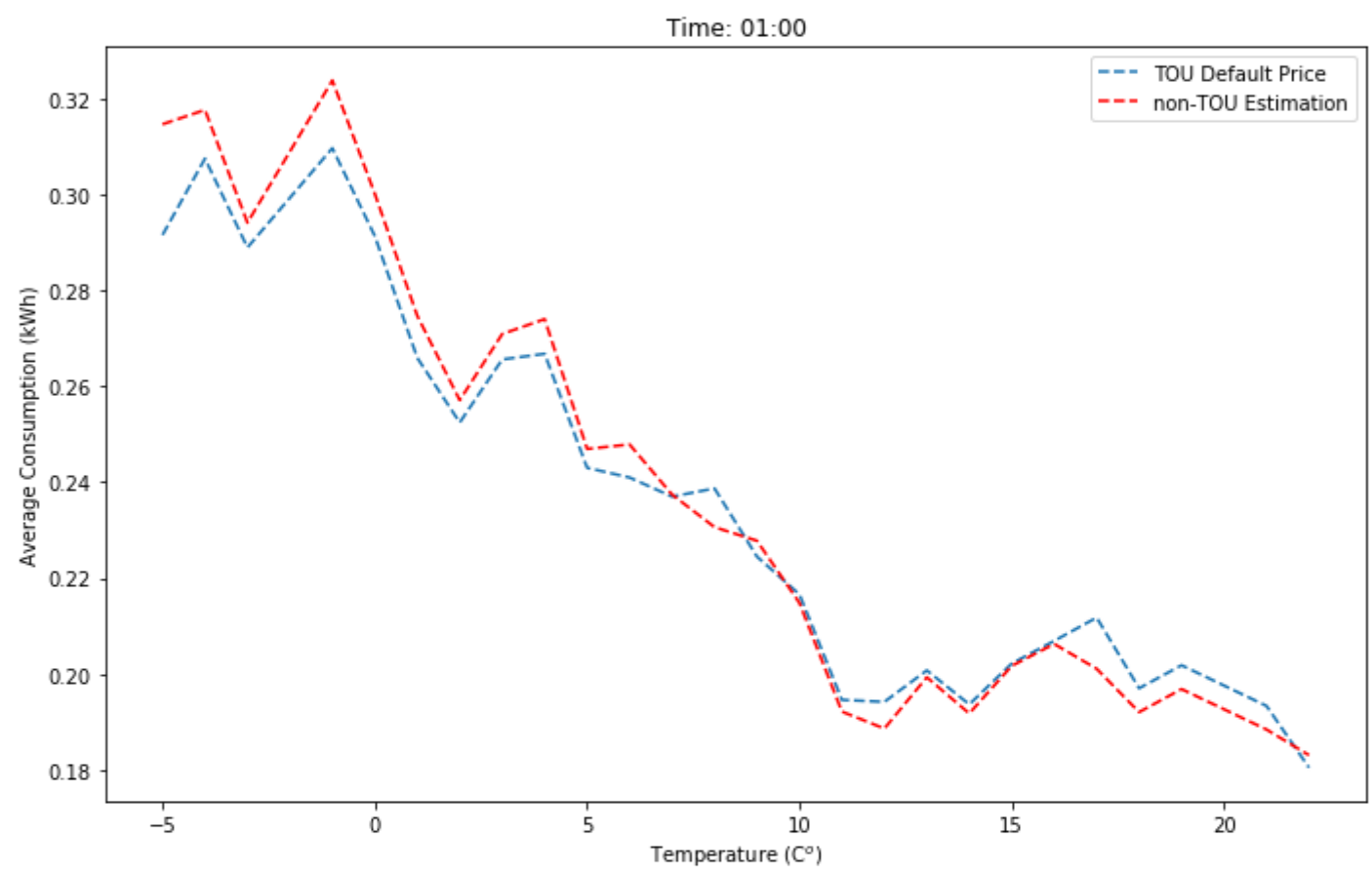
In [117]:
```
df_dVSt
```

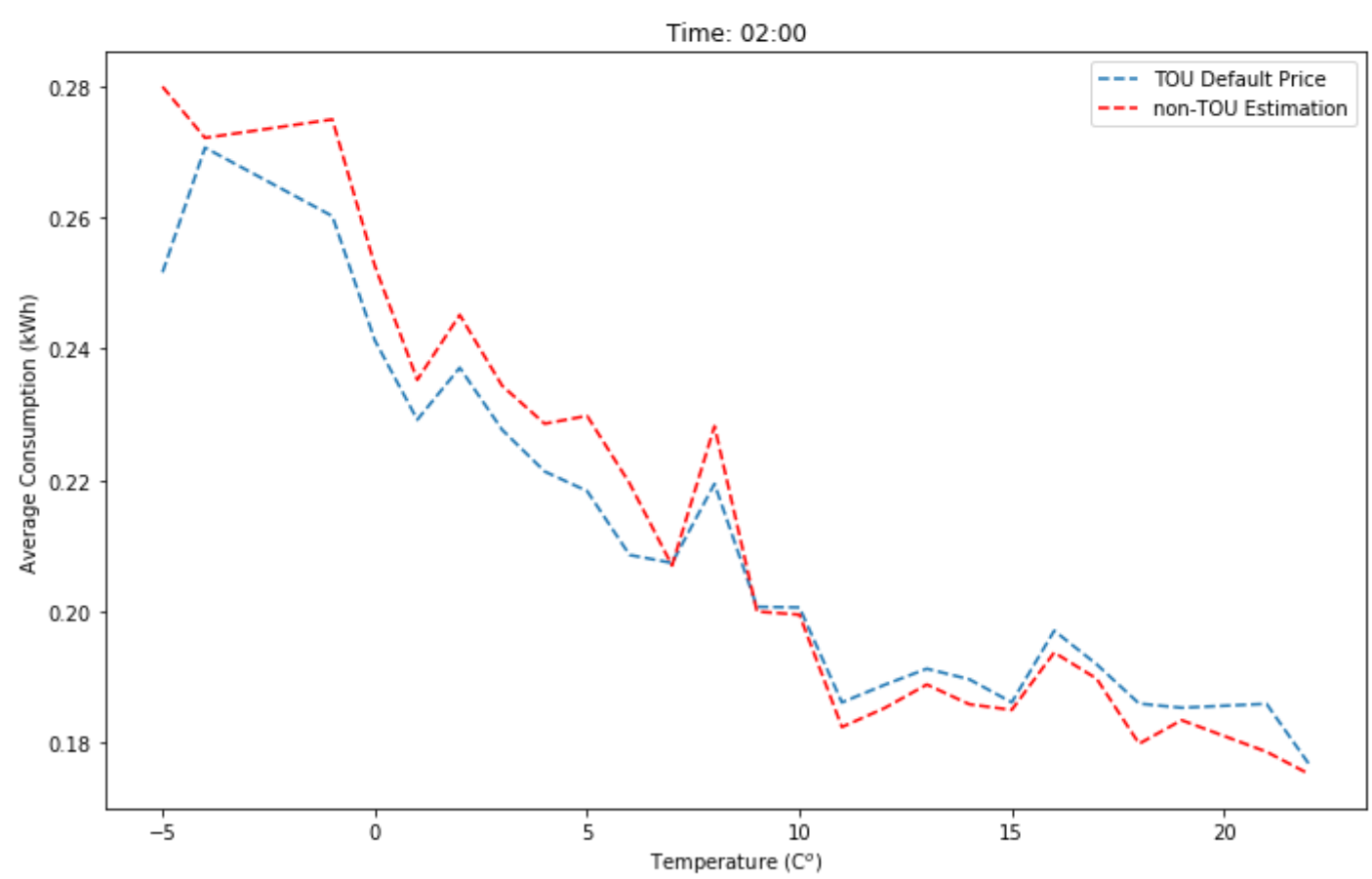
Out[117]:

	TempC	Total
0	-4.0	0.532086
1	-3.0	0.547714
2	-2.0	0.472577
3	0.0	0.524919
4	1.0	0.501798
5	2.0	0.444121
6	3.0	0.467437
7	4.0	0.435415
8	5.0	0.421515
9	6.0	0.398166
10	7.0	0.428464
11	8.0	0.437295
12	9.0	0.395238
13	10.0	0.349187
14	11.0	0.301584
15	12.0	0.298362
16	13.0	0.298521
17	14.0	0.289652
18	15.0	0.283978
19	16.0	0.288039
20	17.0	0.283645
21	18.0	0.268602
22	19.0	0.308320
23	20.0	0.278218
24	22.0	0.274390
25	25.0	0.278884

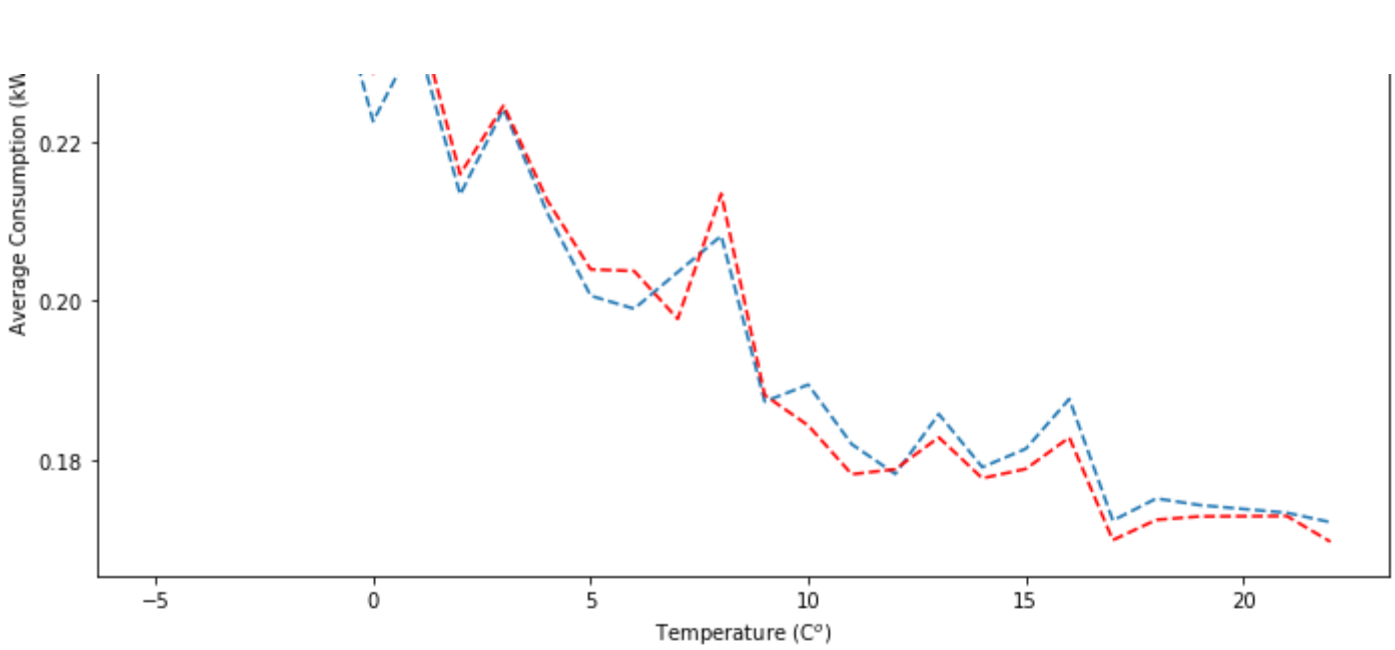
In [115]:
```
# TOU default price vs estimation of TOU group facing default price using non-TOU data
# hourly energy consumption v.s. temperature in event days
fig_all = plt.figure(figsize = (10,150))
ax_Ntou = [] # store subplot objects
ax_tou = [] # store subplot objects
for i in range(24):
    ax_Ntou.append(fig_all.add_subplot(24, 1, i+1))
    if i <= 9:
        # calculate the average difference between two
        # calculate the average consumption grouped by different temperatures
        df_dVSt = df_tou1h_eventday_dPrice[df_tou1h_eventday_dPrice.GMT.str.contains('0' + str(i) + ':00:00'
)] #TOU demand vs temperature for high price periods
        df_dVSt = df_dVSt.groupby('TempC')['Total'].mean() / (df_dVSt.shape[1] - 4)
        df_dVSt = df_dVSt.reset_index()
        ax_Ntou[-1].plot(df_dVSt.TempC, df_dVSt.Total,  c = 'C0', label = 'TOU Default Price', linestyle='das
hed' )

        df_NdVSt_help1 = df_Ntou1h[df_help['GMT'].isin(event_days)]
        df_help2 = df_tariff_1h[df_help['GMT'].isin(event_days)]
        df_NdVSt_help = df_NdVSt_help1[df_help2.Price == 0.1176]
        df_NdVSt = df_NdVSt_help[df_NdVSt_help.GMT.str.contains('0' + str(i) + ':00:00')] # estimation of TOU
group facing default price using non-TOU data (will remove the const difference later)
        df_NdVSt = df_NdVSt.groupby('TempC')['Total'].mean() / (df_NdVSt.shape[1] - 4)
        df_NdVSt = df_NdVSt.reset_index()
        ax_Ntou[-1].plot(df_NdVSt.TempC, df_NdVSt.Total - const_shift[i], c = 'red', label = 'non-TOU Estimat
ion', linestyle='dashed' )

        ax_Ntou[-1].set_title('Time: 0' + str(i) + ':00')
        ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_Ntou[-1].set_ylabel('Average Consumption (kWh)')
        ax_Ntou[-1].legend()
    else:
        df_dVSt = df_tou1h_eventday_dPrice[df_tou1h_eventday_dPrice.GMT.str.contains(str(i) + ':00:00')] #TOU
demand vs temperature for high price periods
        df_dVSt = df_dVSt.groupby('TempC')['Total'].mean() / (df_dVSt.shape[1] - 4)
        df_dVSt = df_dVSt.reset_index()
        ax_Ntou[-1].plot(df_dVSt.TempC, df_dVSt.Total,  c = 'C0', label = 'TOU Default Price', linestyle='das
hed')

        df_NdVSt_help1 = df_Ntou1h[df_help['GMT'].isin(event_days)]
        df_help2 = df_tariff_1h[df_help['GMT'].isin(event_days)]
        df_NdVSt_help = df_NdVSt_help1[df_help2.Price == 0.1176]
        df_NdVSt = df_NdVSt_help[df_NdVSt_help.GMT.str.contains(str(i) + ':00:00')] # estimation of TOU group
facing default price using non-TOU data (will remove the const difference later)
        df_NdVSt = df_NdVSt.groupby('TempC')['Total'].mean() / (df_NdVSt.shape[1] - 4)
        df_NdVSt = df_NdVSt.reset_index()
        ax_Ntou[-1].plot(df_NdVSt.TempC, df_NdVSt.Total - const_shift[i], c = 'red', label = 'non-TOU Estimat
ion', linestyle='dashed' )

        ax_Ntou[-1].set_title('Time: ' + str(i) + ':00')
        ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_Ntou[-1].set_ylabel('Average Consumption (kWh)')
        ax_Ntou[-1].legend()
plt.tight_layout()
```

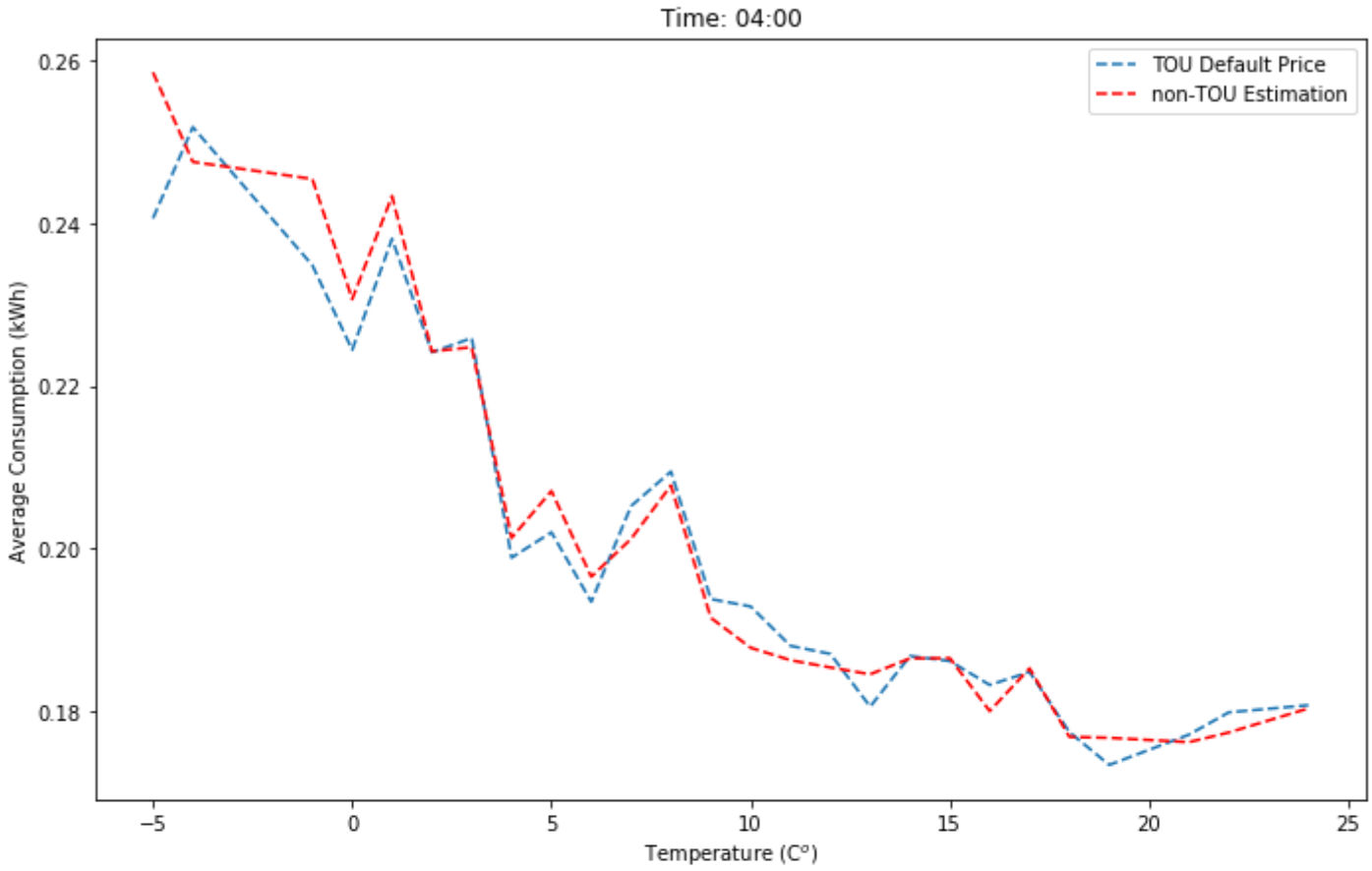

Time: 00:00
TOU Default Price
non-TOU Estimation
Average Consumption (kWh)
Temperature (C°)

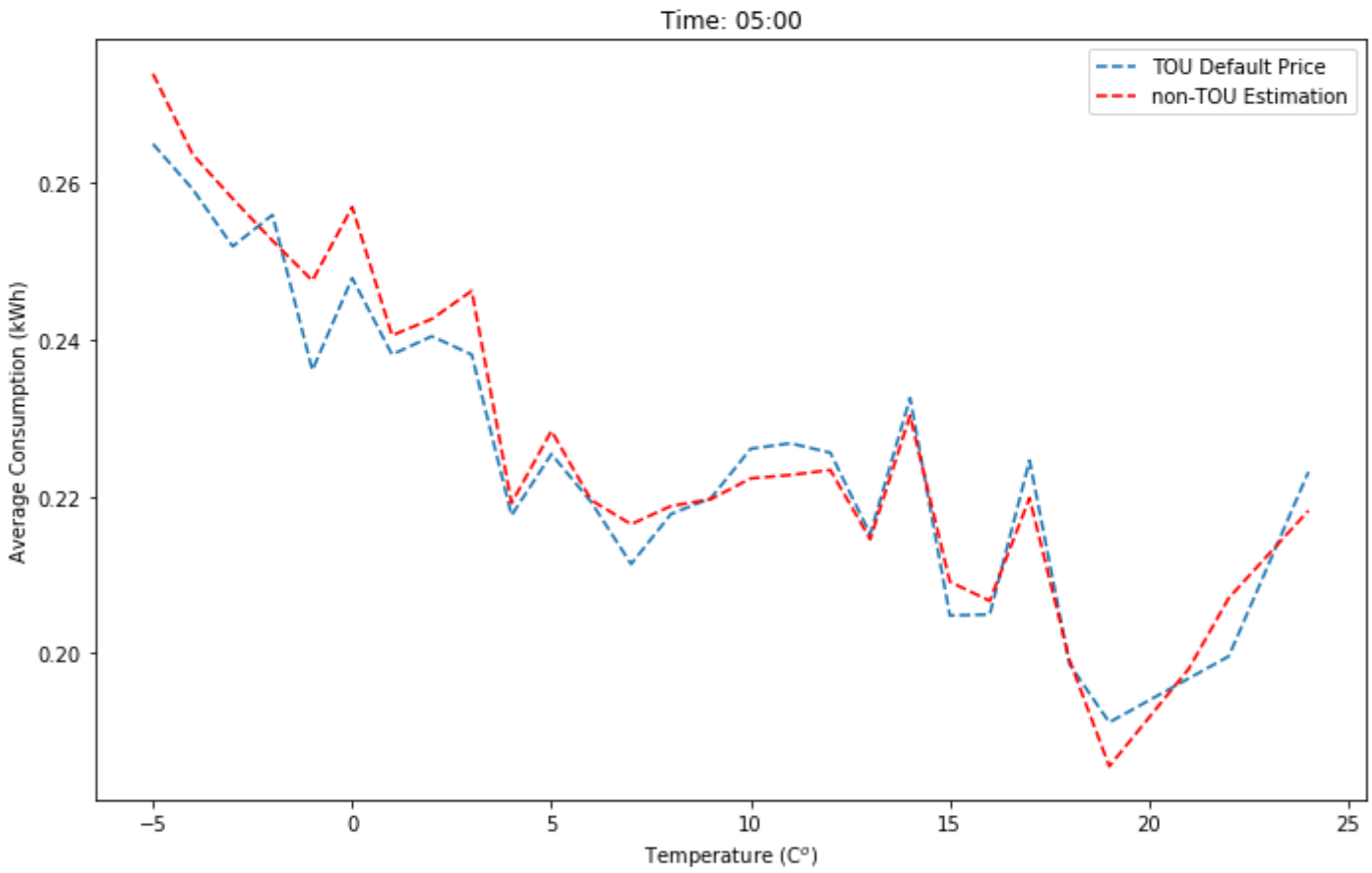

Time: 01:00
TOU Default Price
non-TOU Estimation
Average Consumption (kWh)
Temperature (C°)

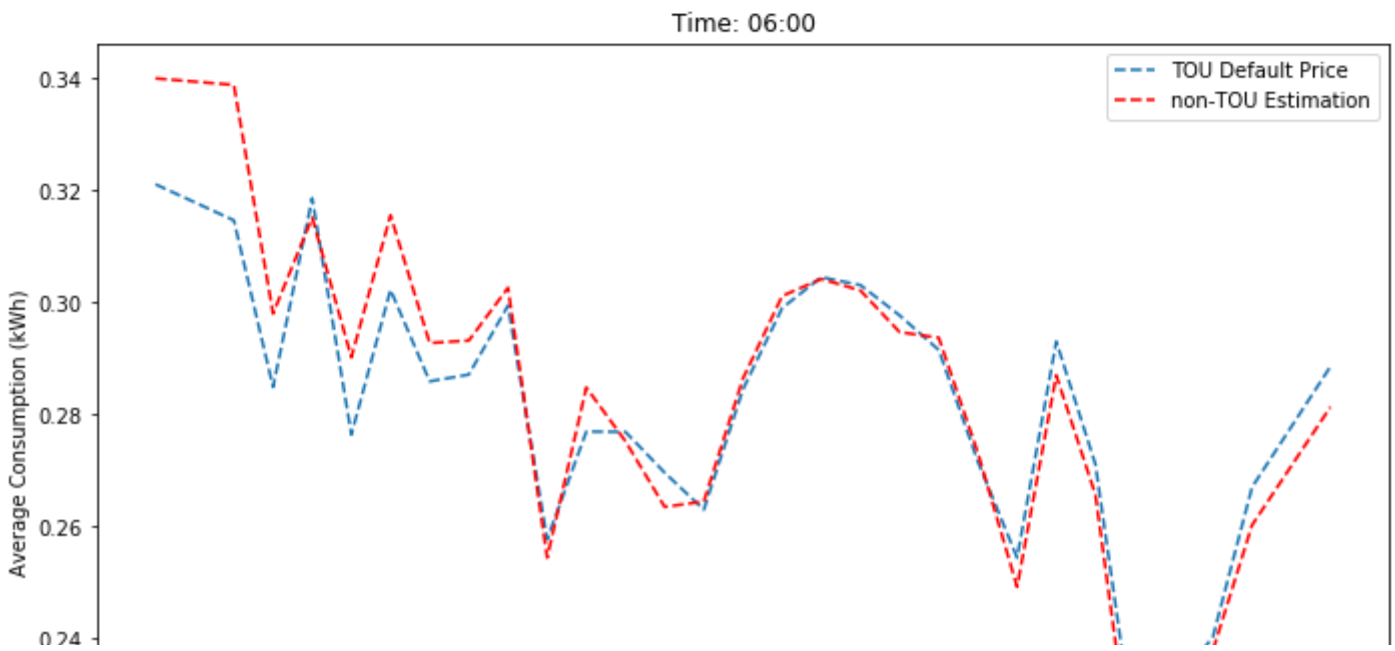

Time: 02:00
TOU Default Price
non-TOU Estimation
Average Consumption (kWh)
Temperature (C°)

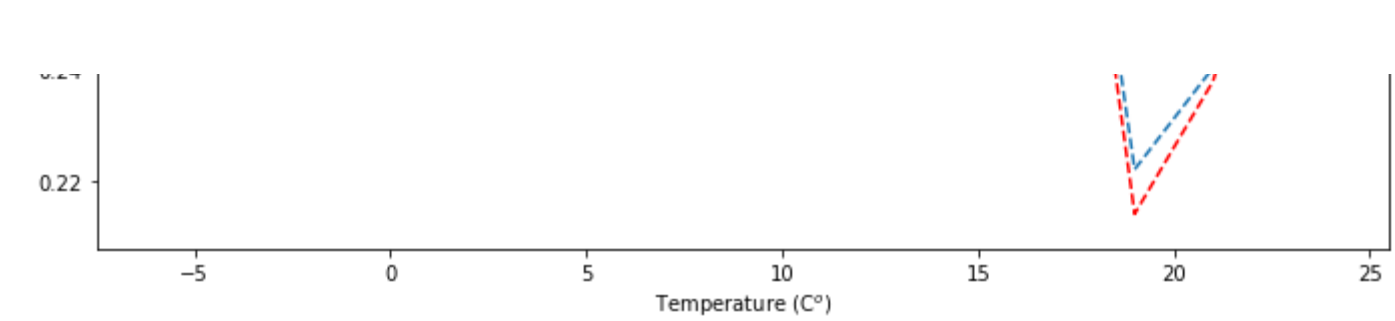

Time: 03:00
TOU Default Price
non-TOU Estimation

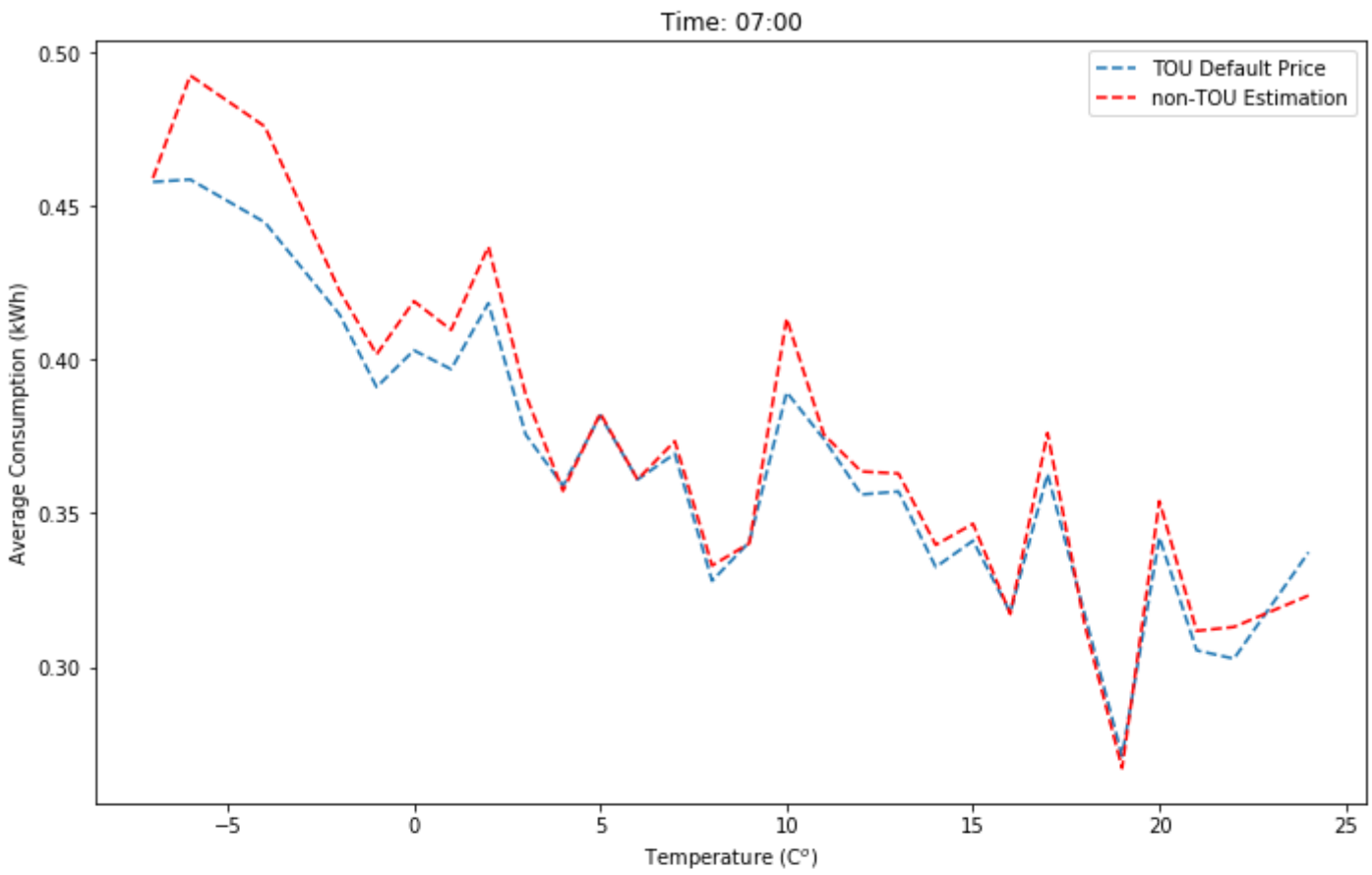

Average Consumption (kW
0.22
0.20
0.18
−5
0
5
10
15
20
Temperature (C°)

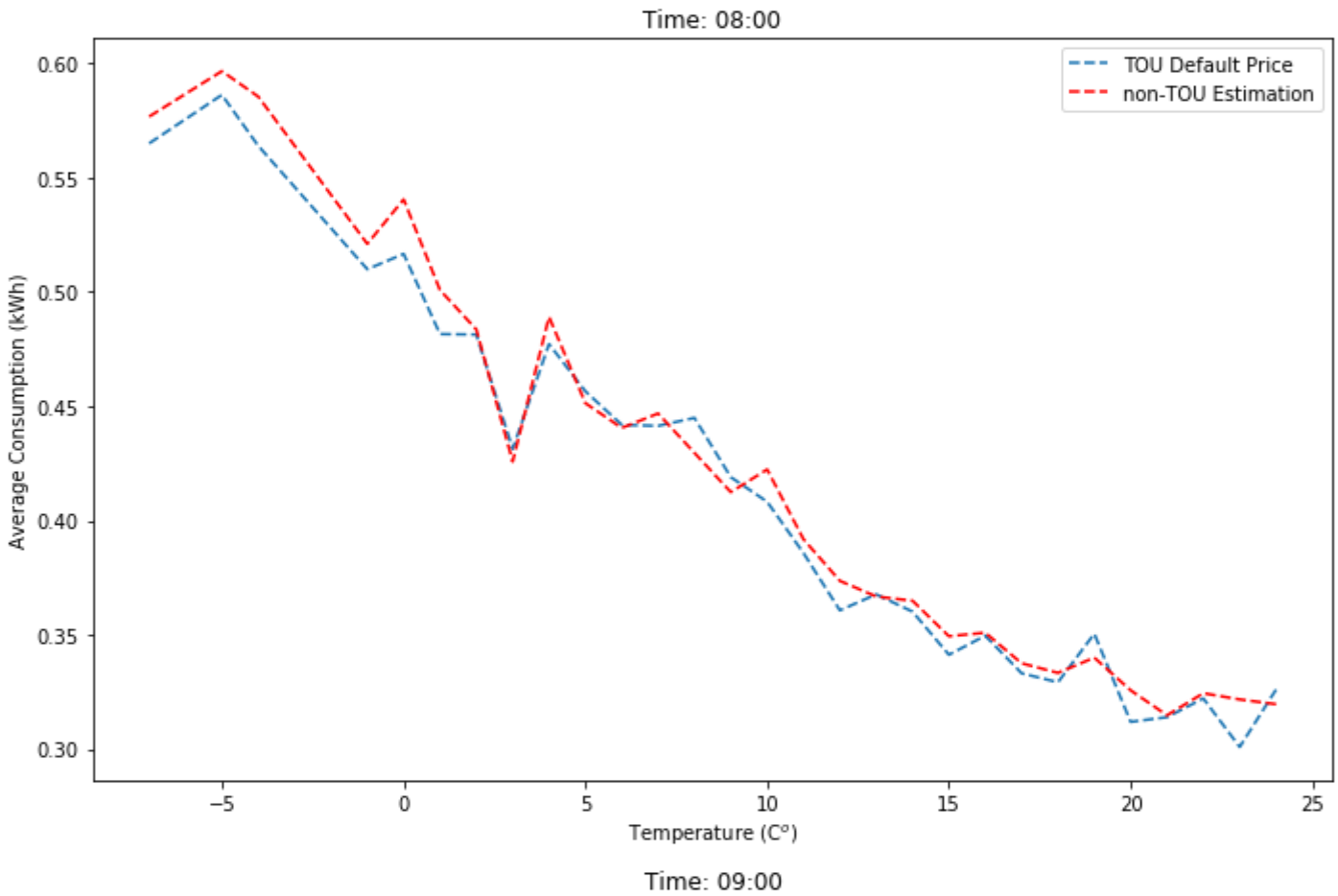

Time: 04:00
TOU Default Price
non-TOU Estimation
Average Consumption (kWh)
0.26
0.24
0.22
0.20
0.18
−5
0
5
10
15
20
25
Temperature (C°)

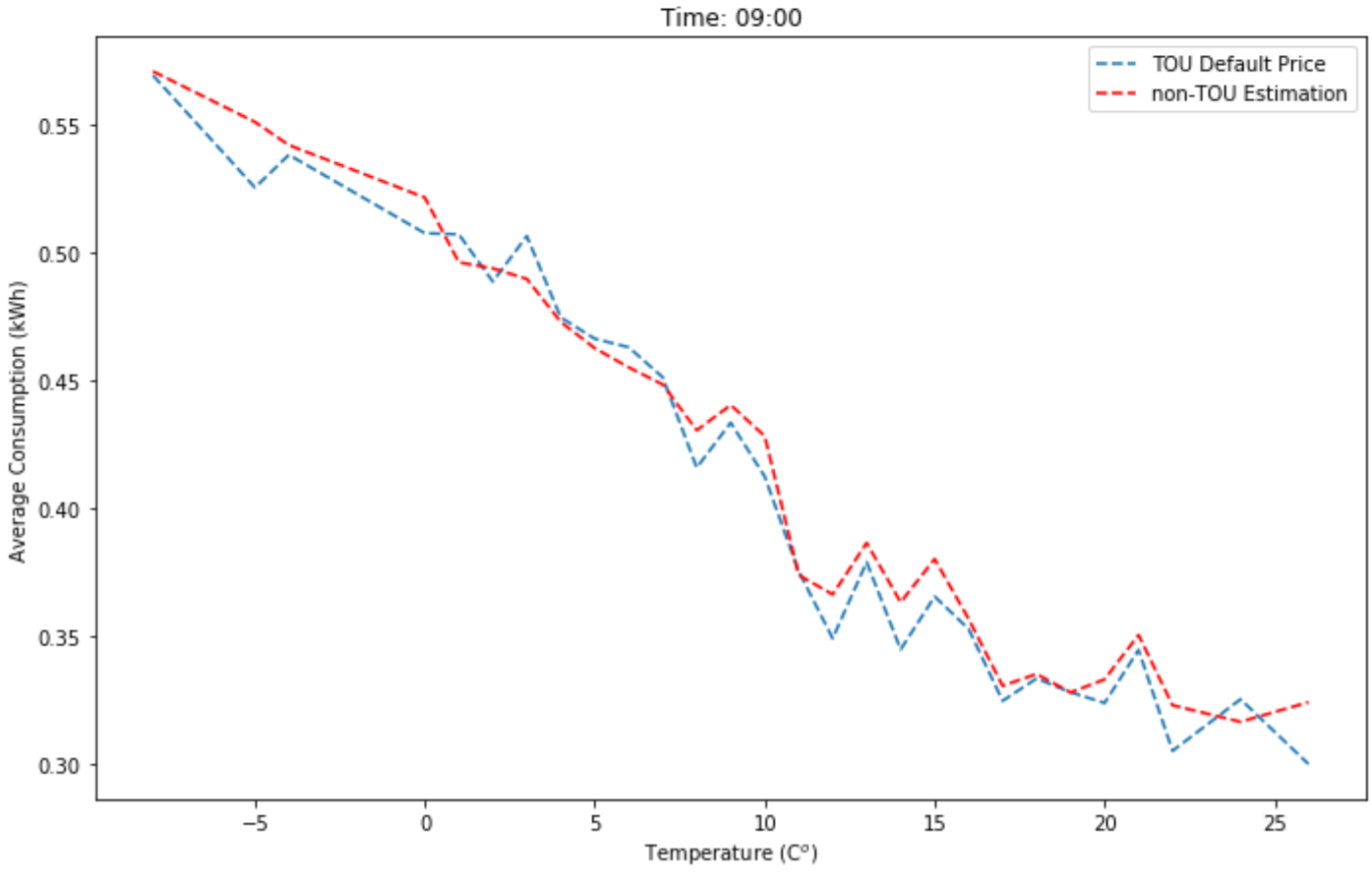

Time: 05:00
TOU Default Price
non-TOU Estimation
Average Consumption (kWh)
0.26
0.24
0.22
0.20
−5
0
5
10
15
20
25
Temperature (C°)


Time: 06:00
TOU Default Price
non-TOU Estimation
Average Consumption (kWh)
0.34
0.32
0.30
0.28
0.26
0.24

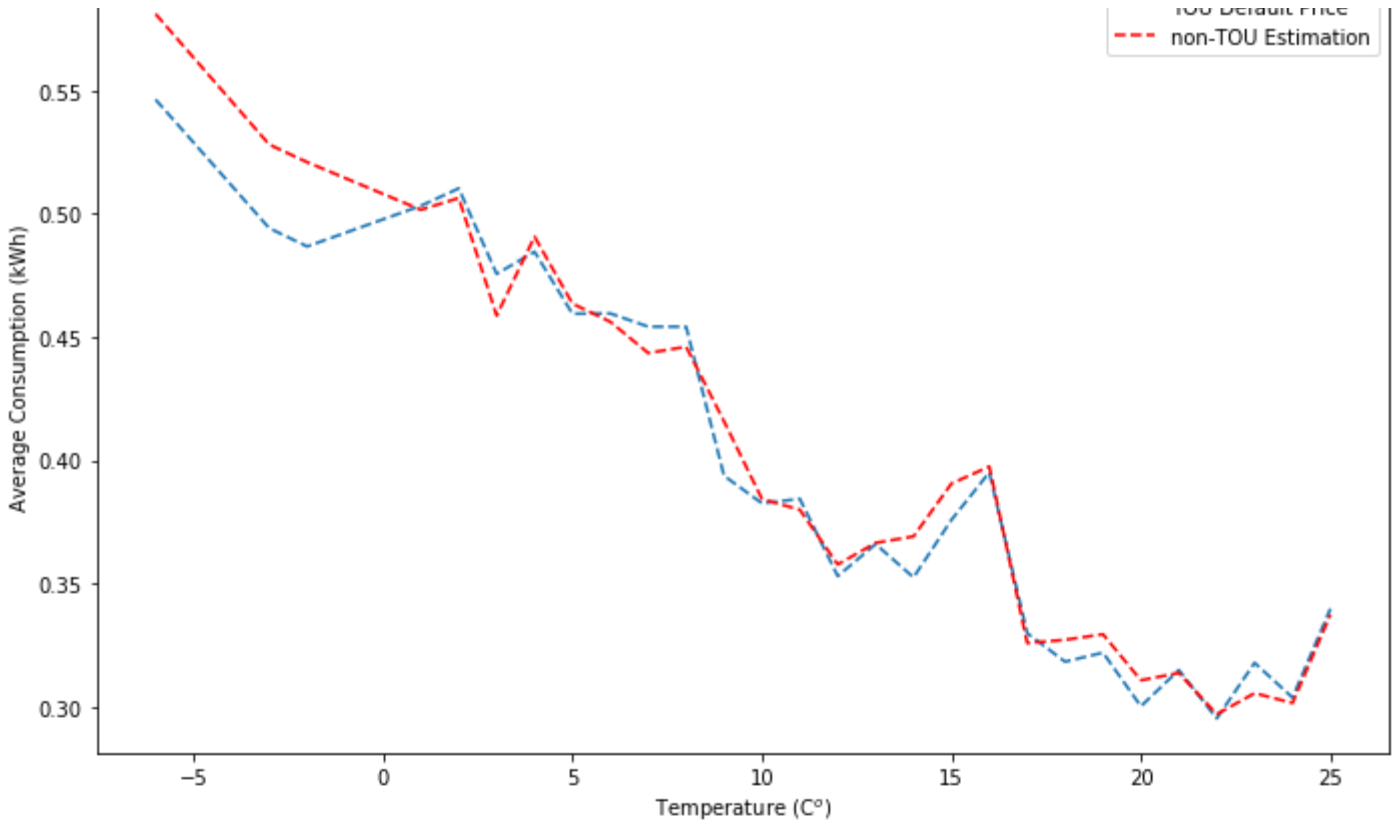

0.22
−5
0
5
10
15
20
25
Temperature (C°)


Time: 07:00
TOU Default Price
non-TOU Estimation
Average Consumption (kWh)
0.50
0.45
0.40
0.35
0.30
−5
0
5
10
15
20
25
Temperature (C°)

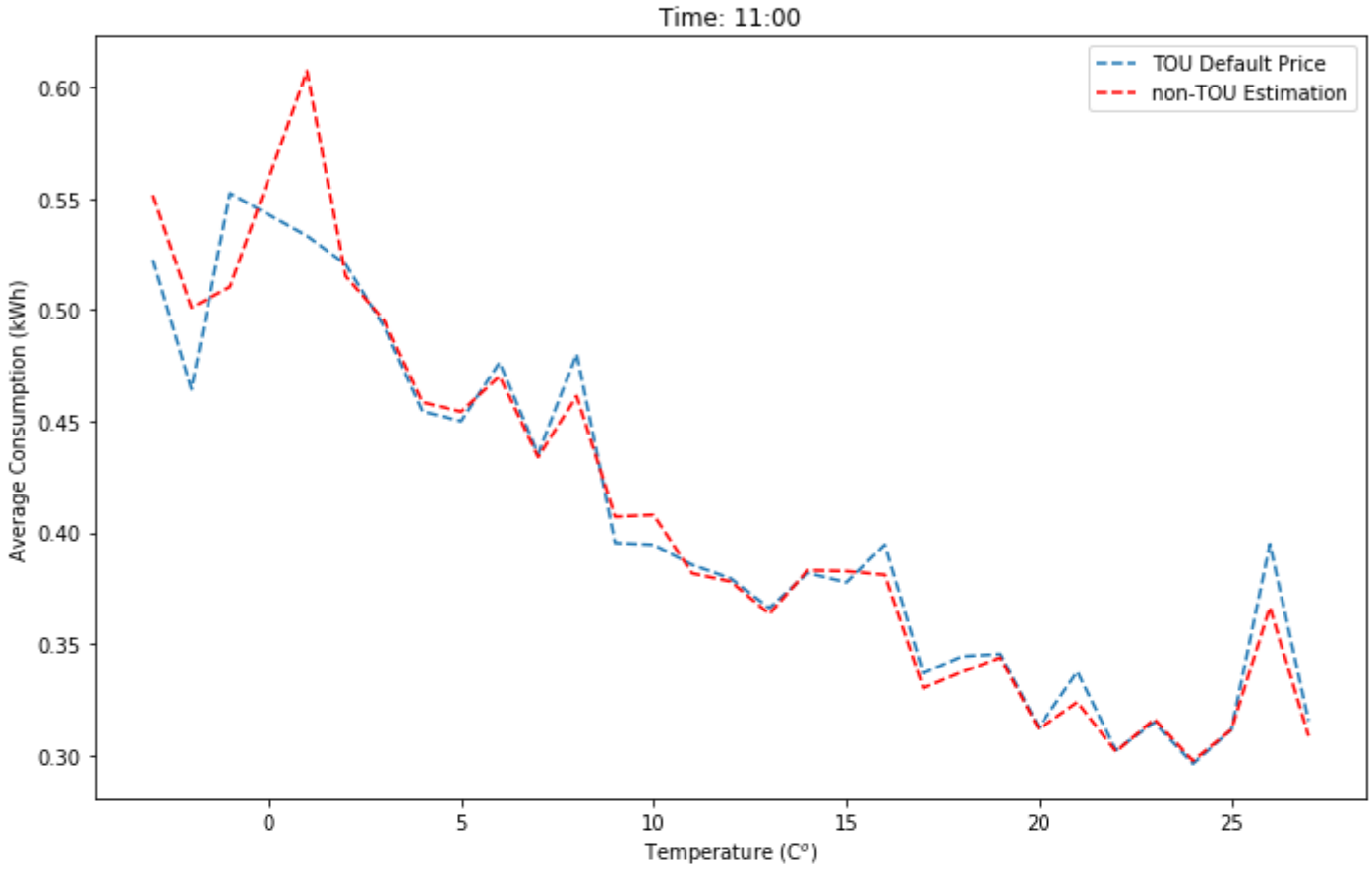

Time: 08:00
TOU Default Price
non-TOU Estimation
Average Consumption (kWh)
0.60
0.55
0.50
0.45
0.40
0.35
0.30
−5
0
5
10
15
20
25
Temperature (C°)


Time: 09:00
TOU Default Price
non-TOU Estimation
Average Consumption (kWh)
0.55
0.50
0.45
0.40
0.35
0.30
−5
0
5
10
15
20
25
Temperature (C°)

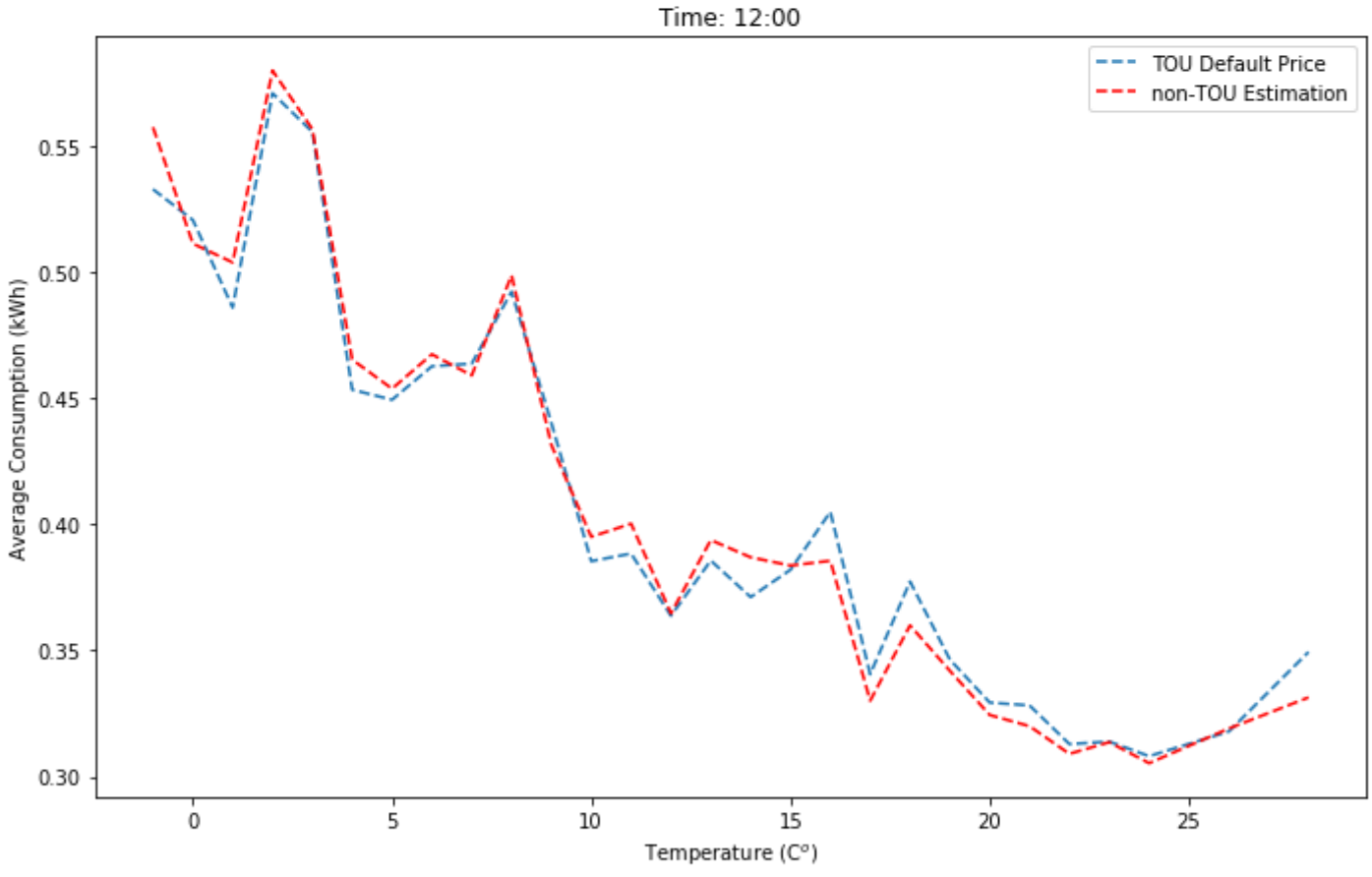
Time: 10:00
TOU Default Price



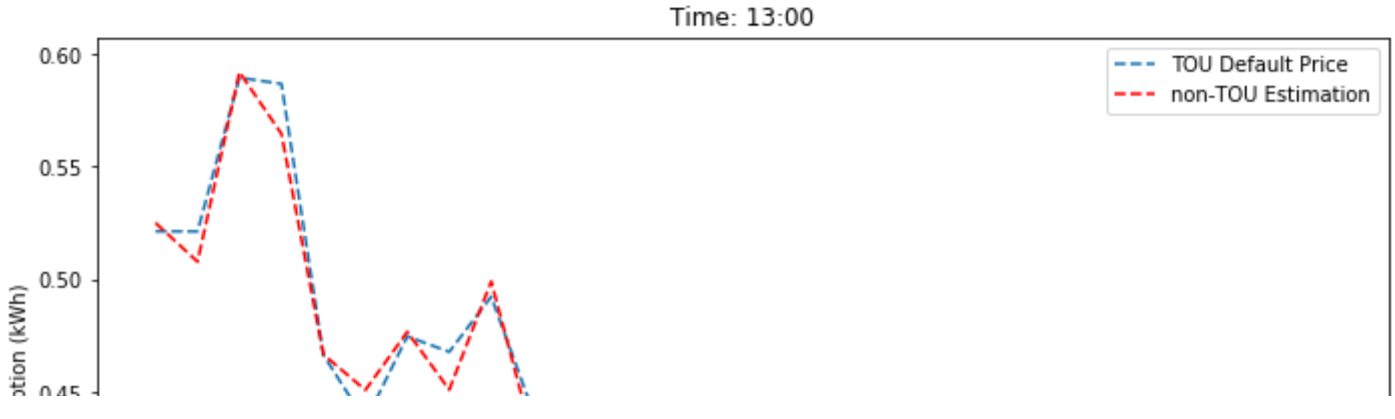
Time: 11:00

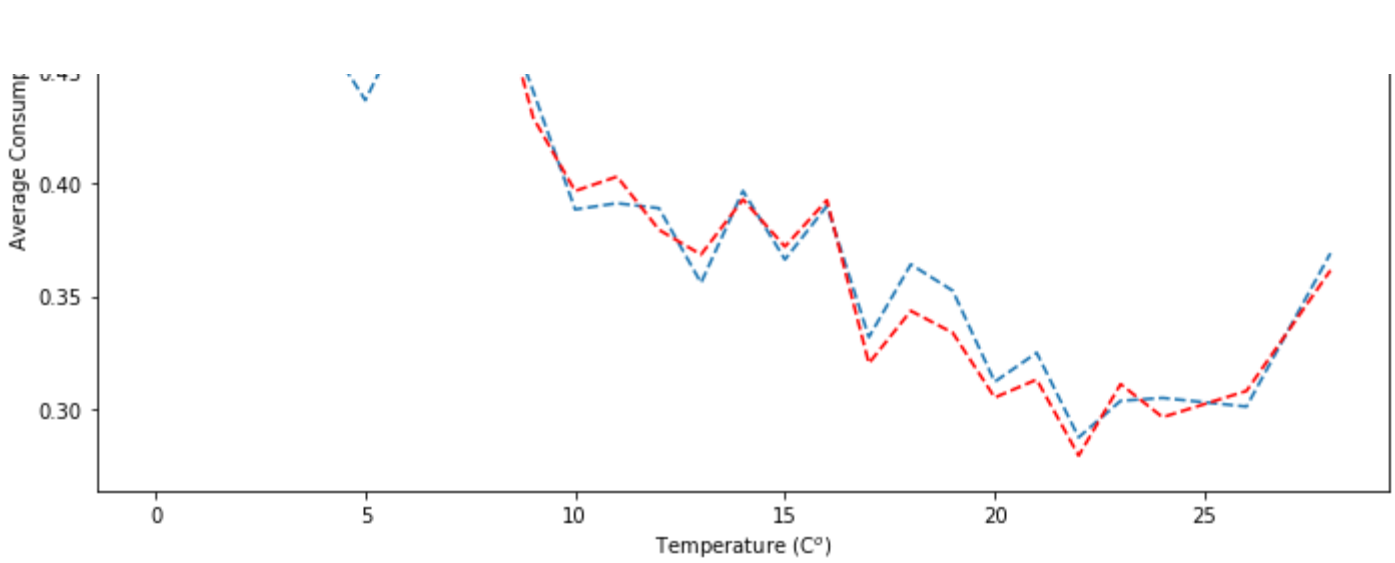


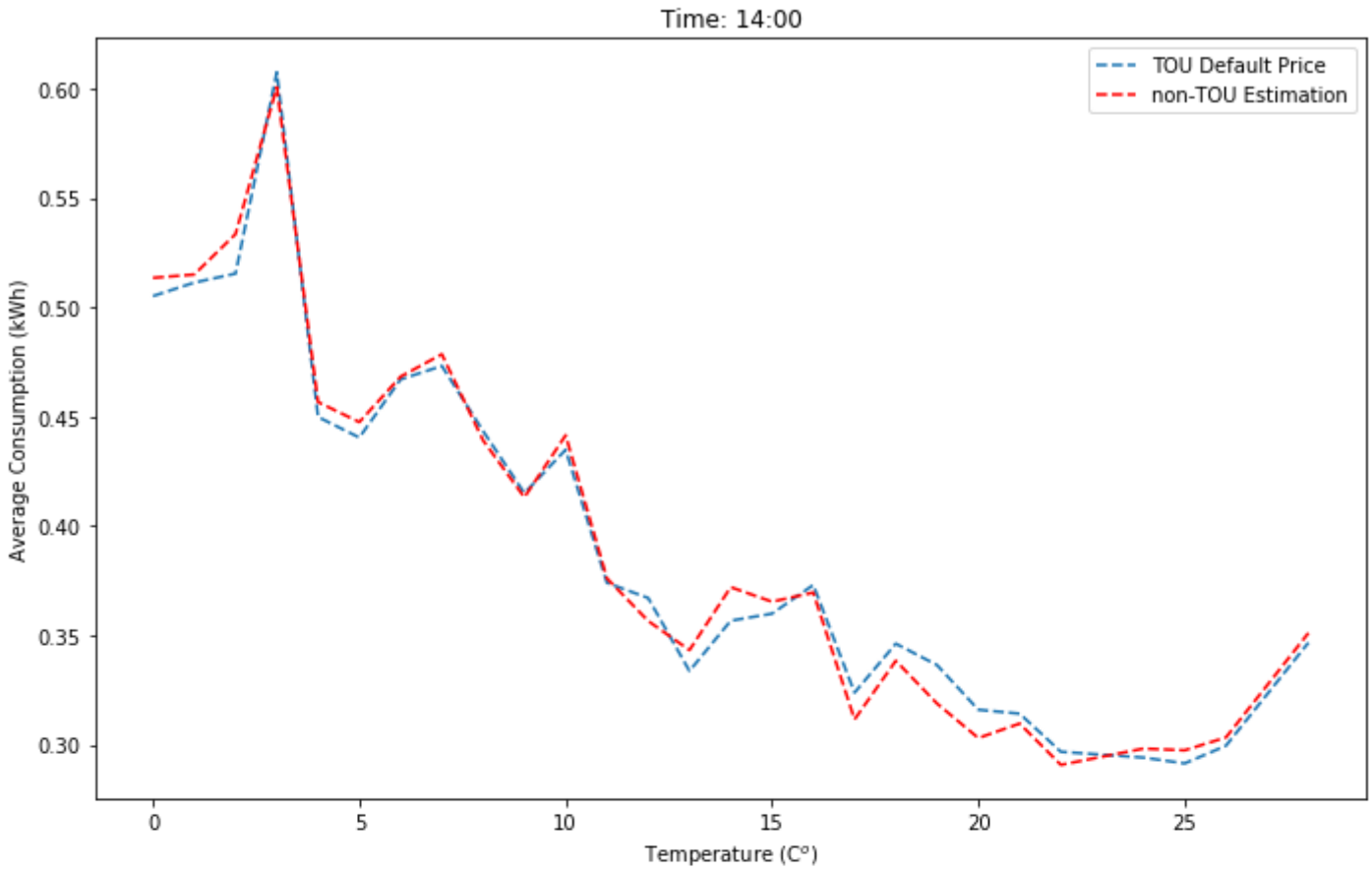
Time: 12:00

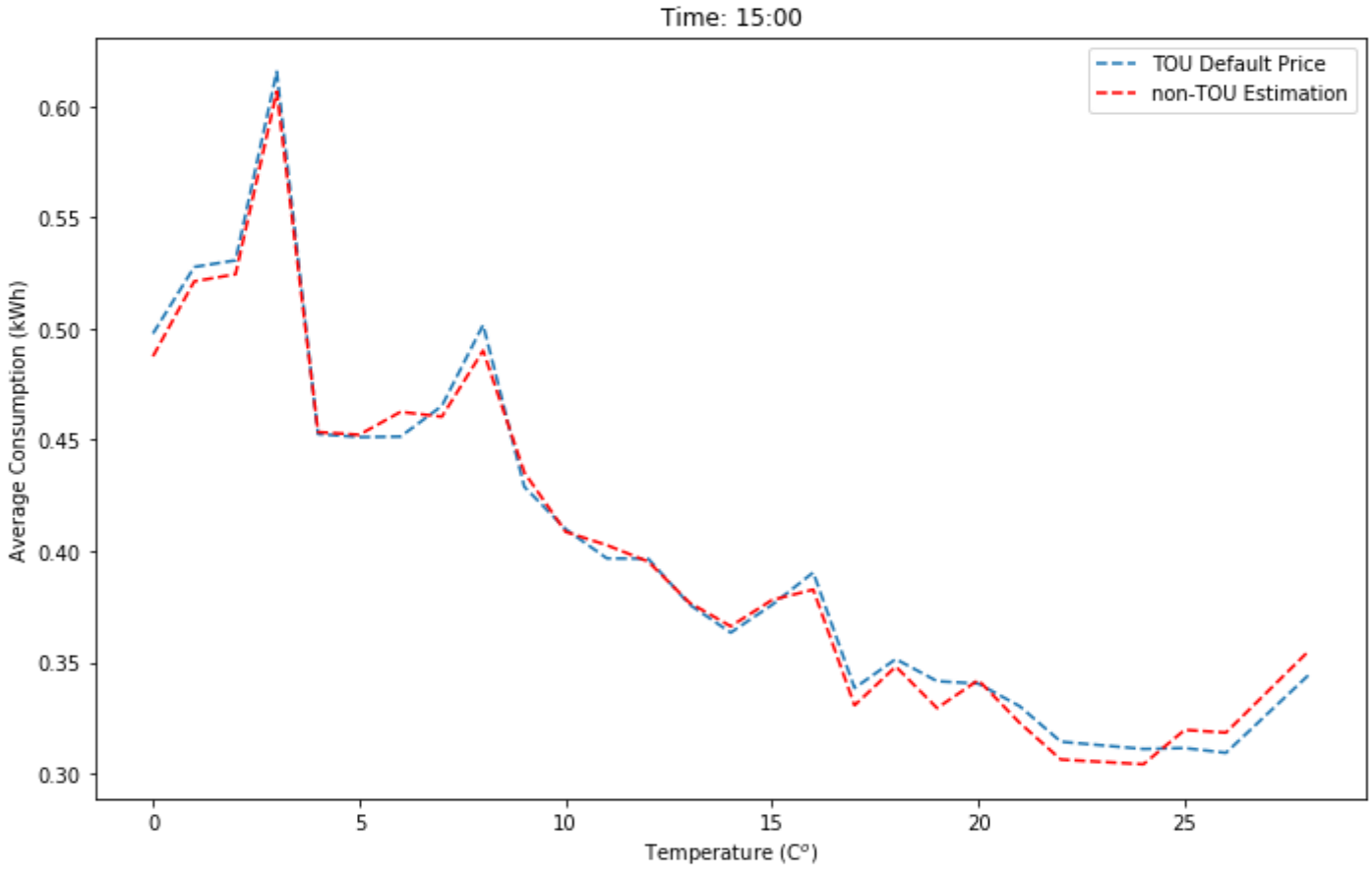


Time: 13:00

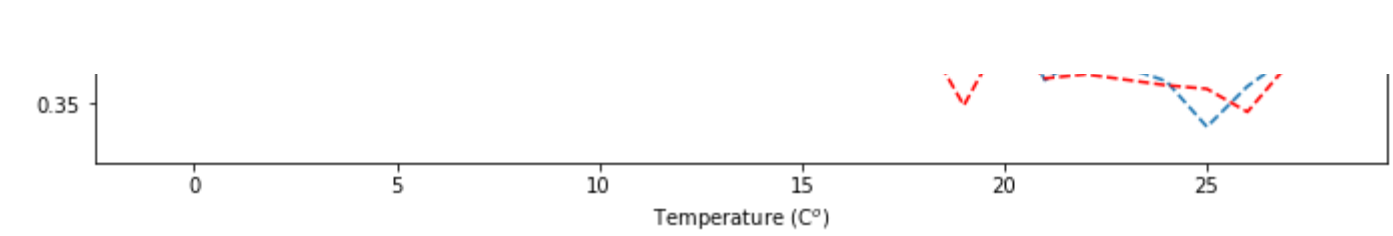


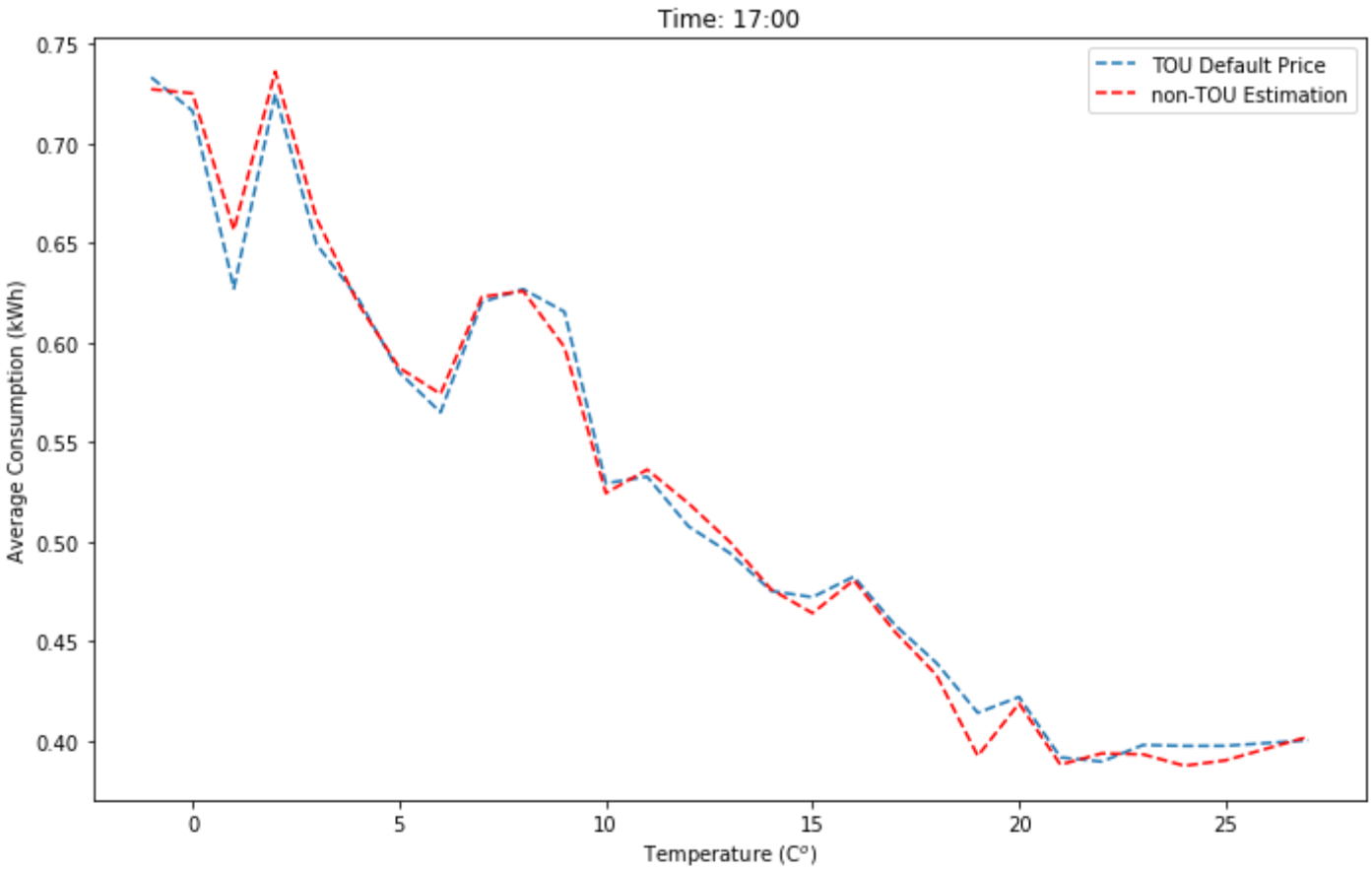

Average Consump
0.40
0.35
0.30
0
5
10
15
20
25
Temperature (C°)

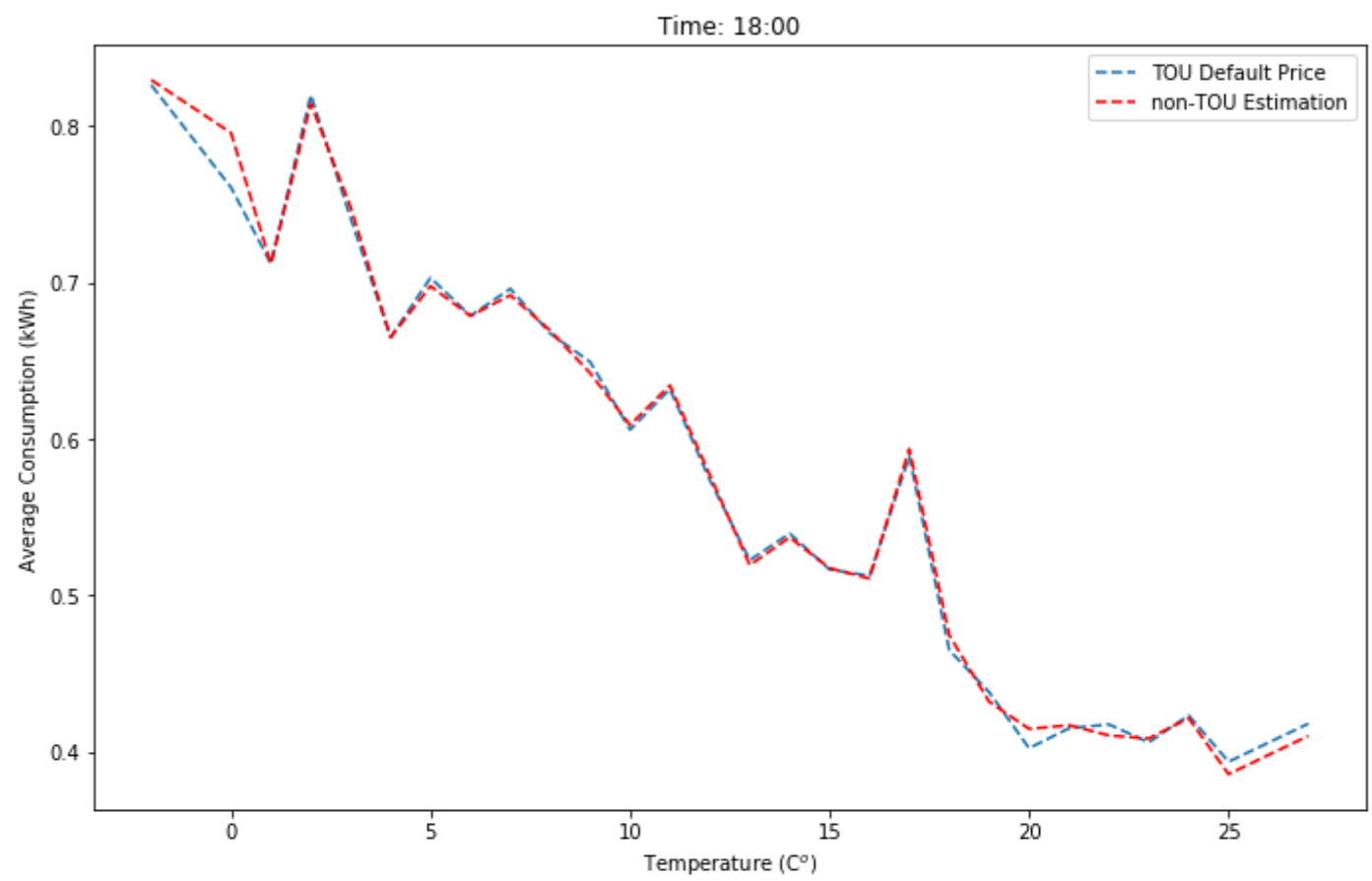

Time: 14:00
TOU Default Price
non-TOU Estimation
Average Consumption (kWh)
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0
5
10
15
20
25
Temperature (C°)

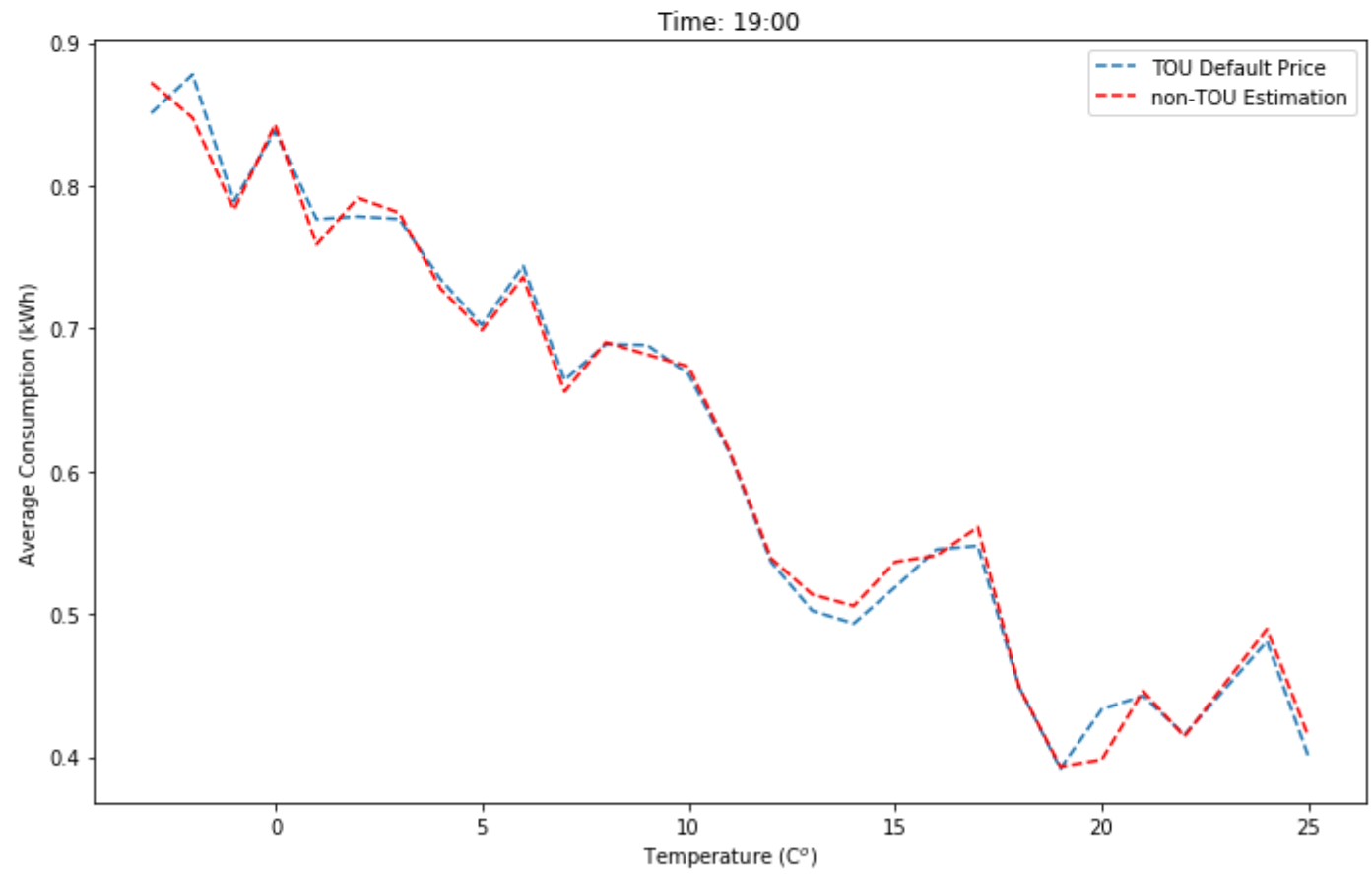

Time: 15:00
TOU Default Price
non-TOU Estimation
Average Consumption (kWh)
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0
5
10
15
20
25
Temperature (C°)

Time: 16:00
TOU Default Price
non-TOU Estimation
Average Consumption (kWh)
0.60
0.55
0.50
0.45
0.40

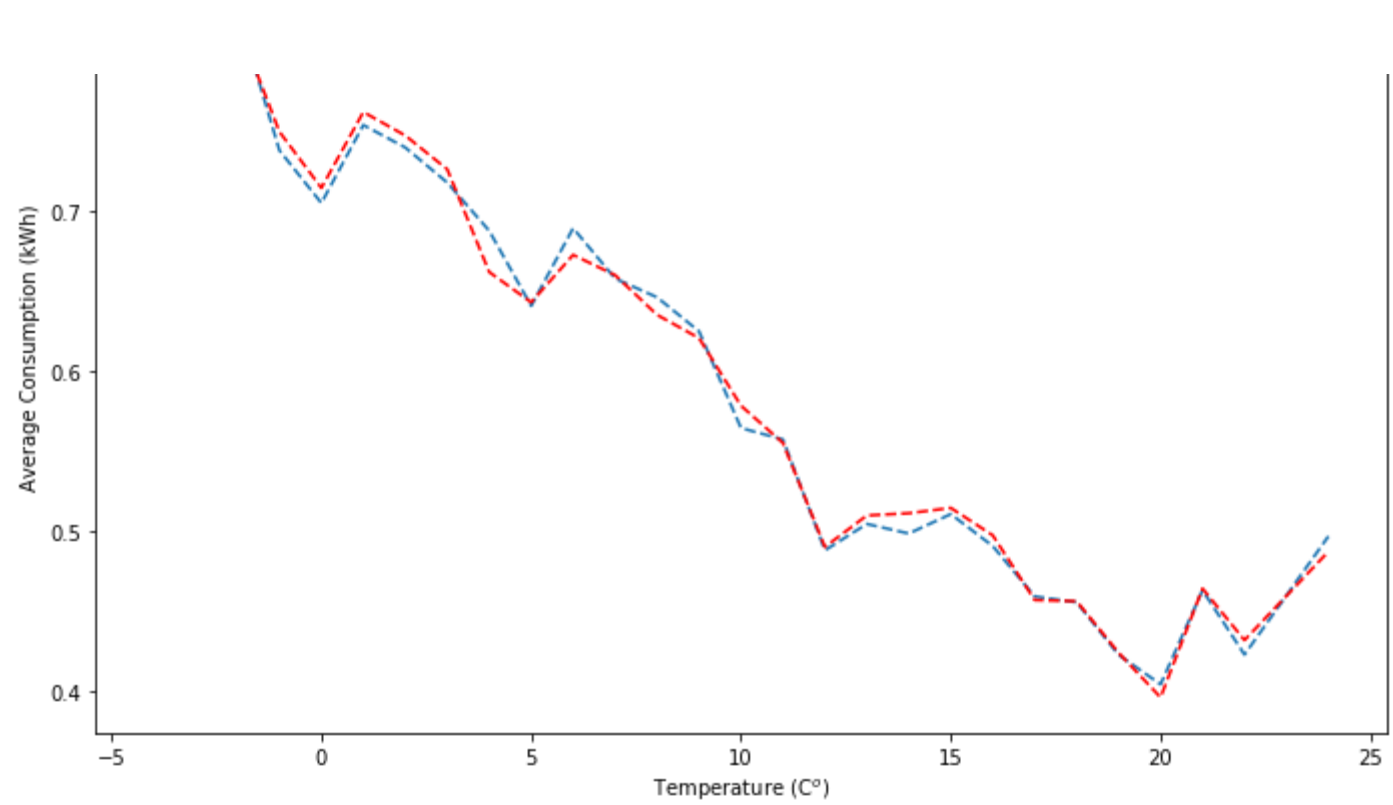

0.35
0
5
10
15
20
25
Temperature (C°)

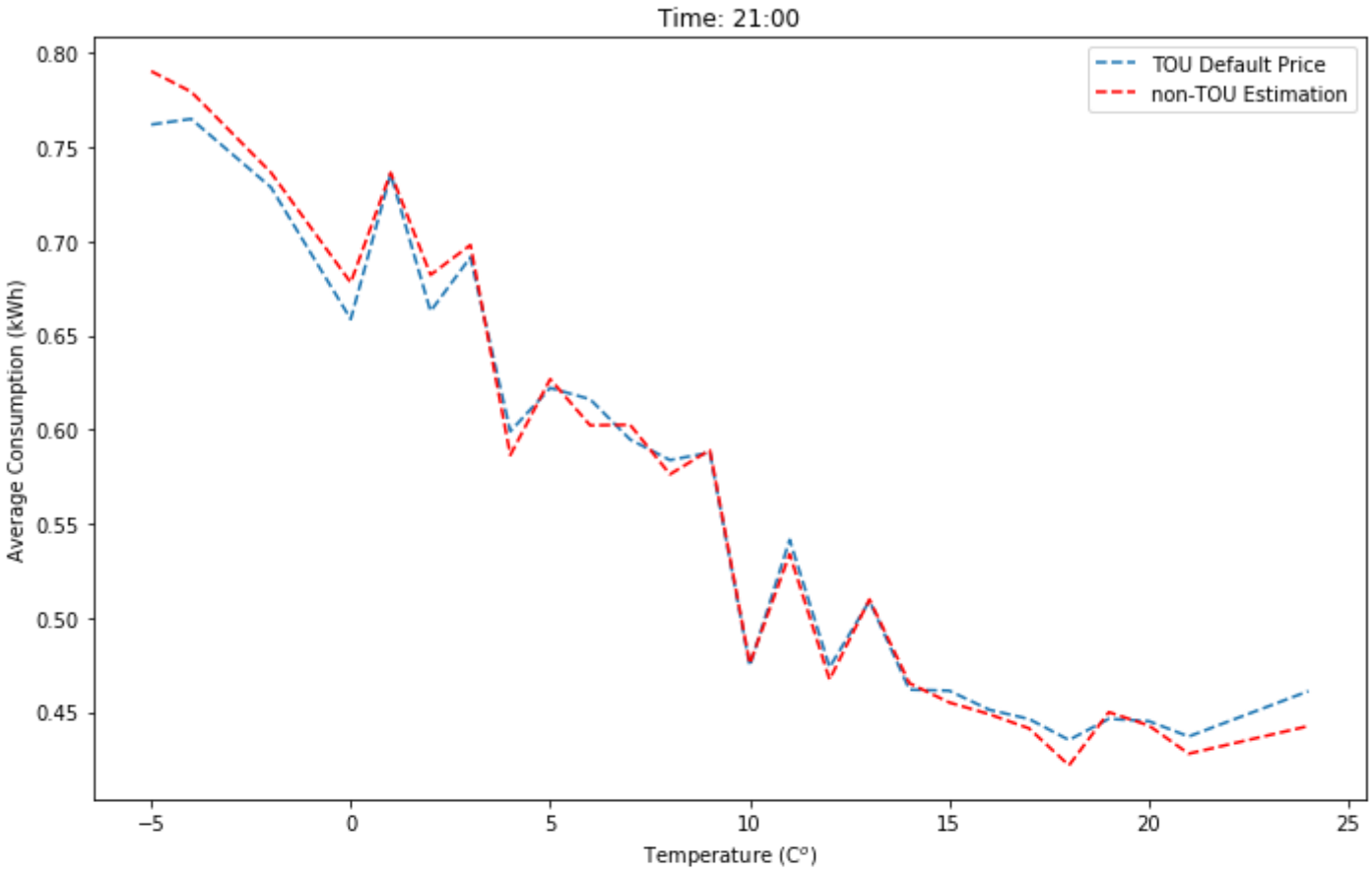

Time: 17:00
TOU Default Price
non-TOU Estimation
Average Consumption (kWh)
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0
5
10
15
20
25
Temperature (C°)

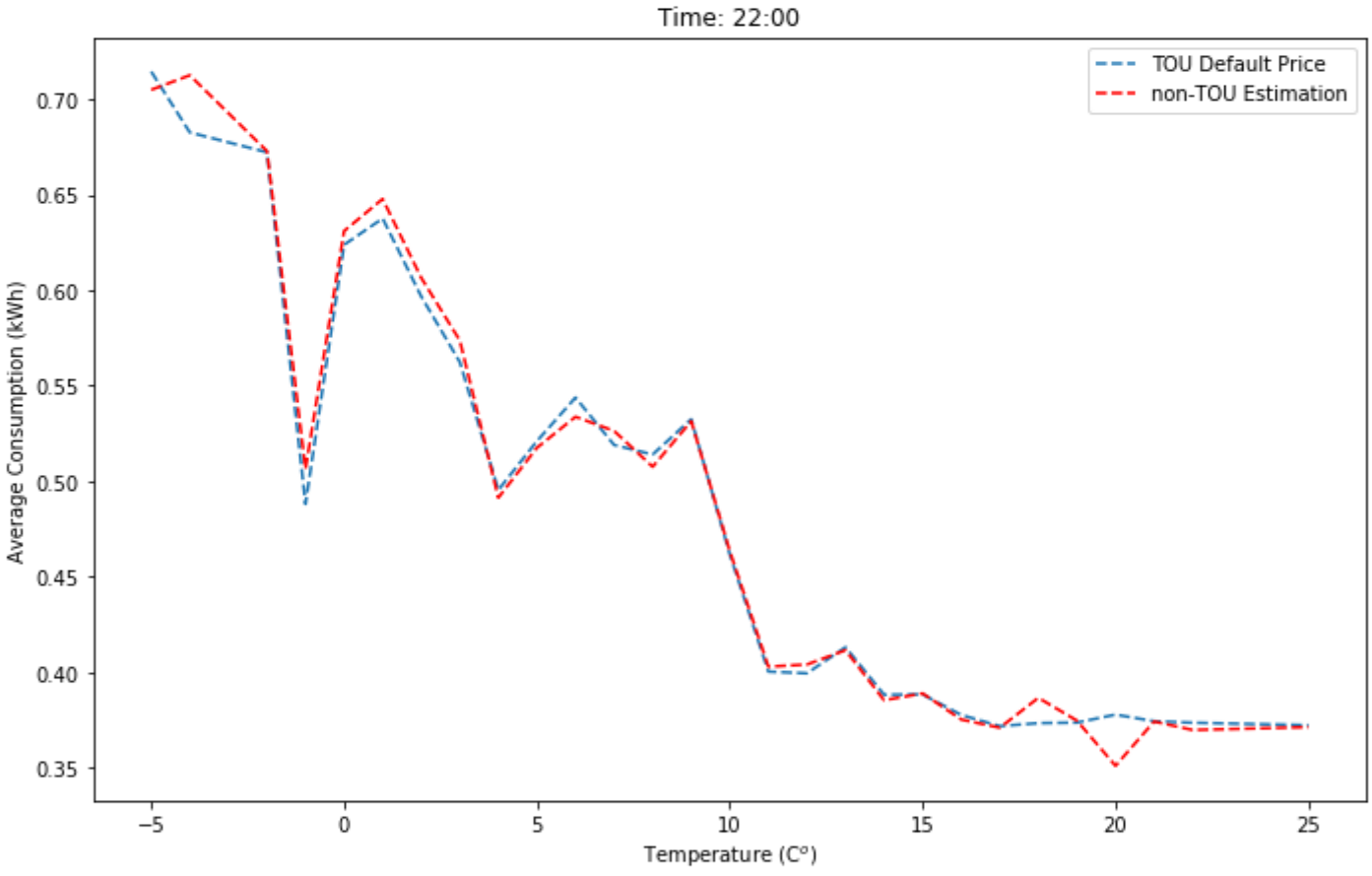

Time: 18:00
TOU Default Price
non-TOU Estimation
Average Consumption (kWh)
0.8
0.7
0.6
0.5
0.4
0
5
10
15
20
25
Temperature (C°)

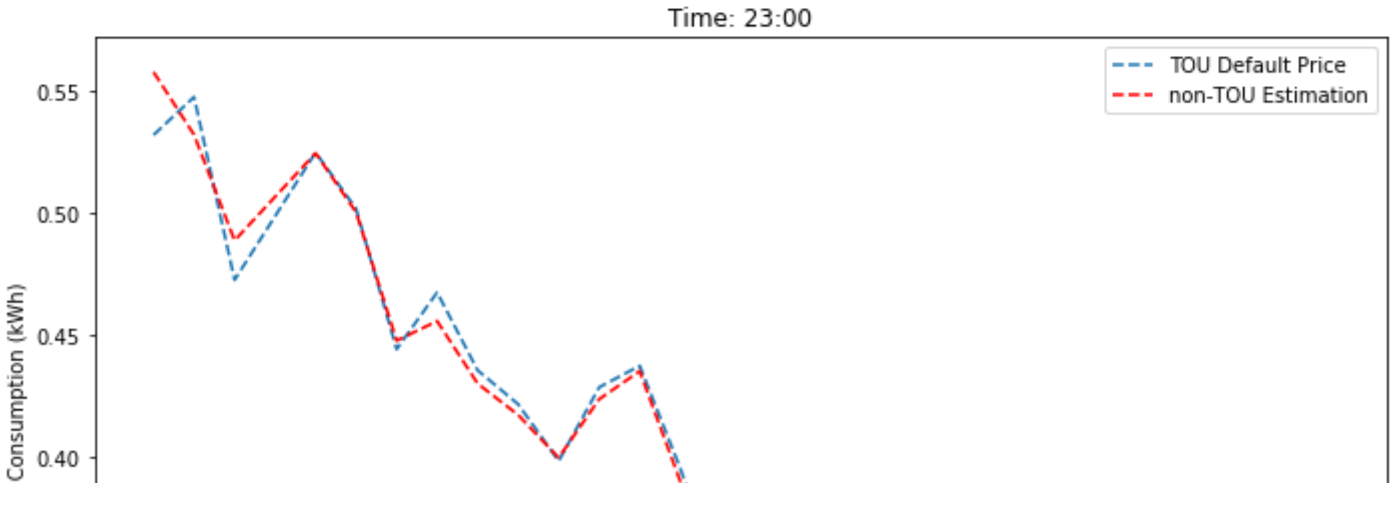

Time: 19:00
TOU Default Price
non-TOU Estimation
Average Consumption (kWh)
0.9
0.8
0.7
0.6
0.5
0.4
0
5
10
15
20
25
Temperature (C°)

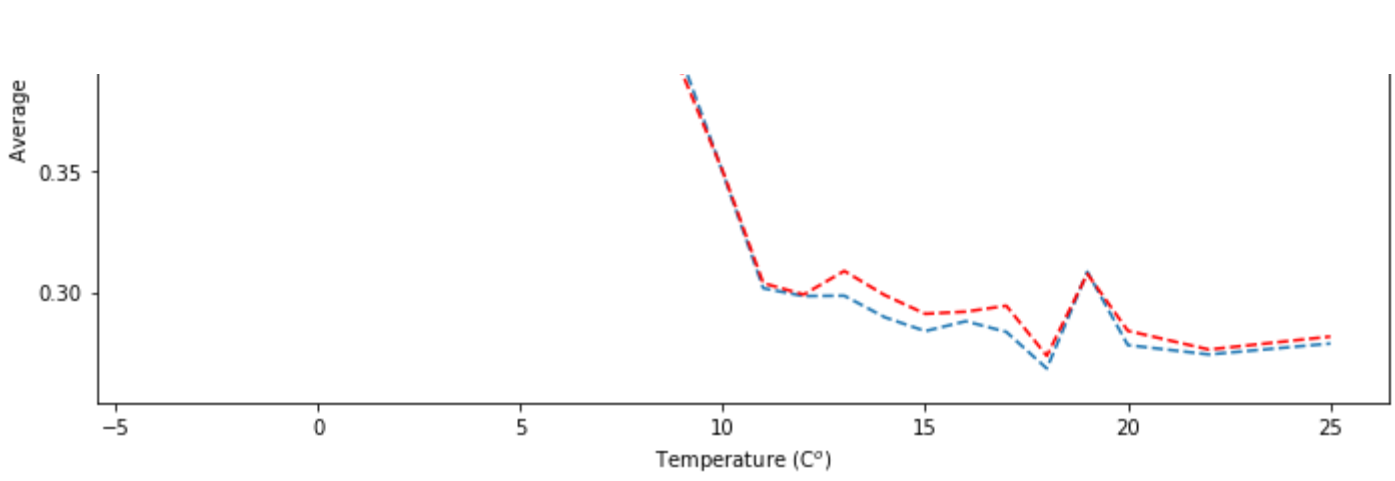

Time: 20:00
TOU Default Price
non-TOU Estimation
0.8


Average Consumption (kWh)
0.7
0.6
0.5
0.4
−5
0
5
10
15
20
25
Temperature (C°)


Time: 21:00
TOU Default Price
non-TOU Estimation
Average Consumption (kWh)
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
−5
0
5
10
15
20
25
Temperature (C°)


Time: 22:00
TOU Default Price
non-TOU Estimation
Average Consumption (kWh)
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
−5
0
5
10
15
20
25
Temperature (C°)

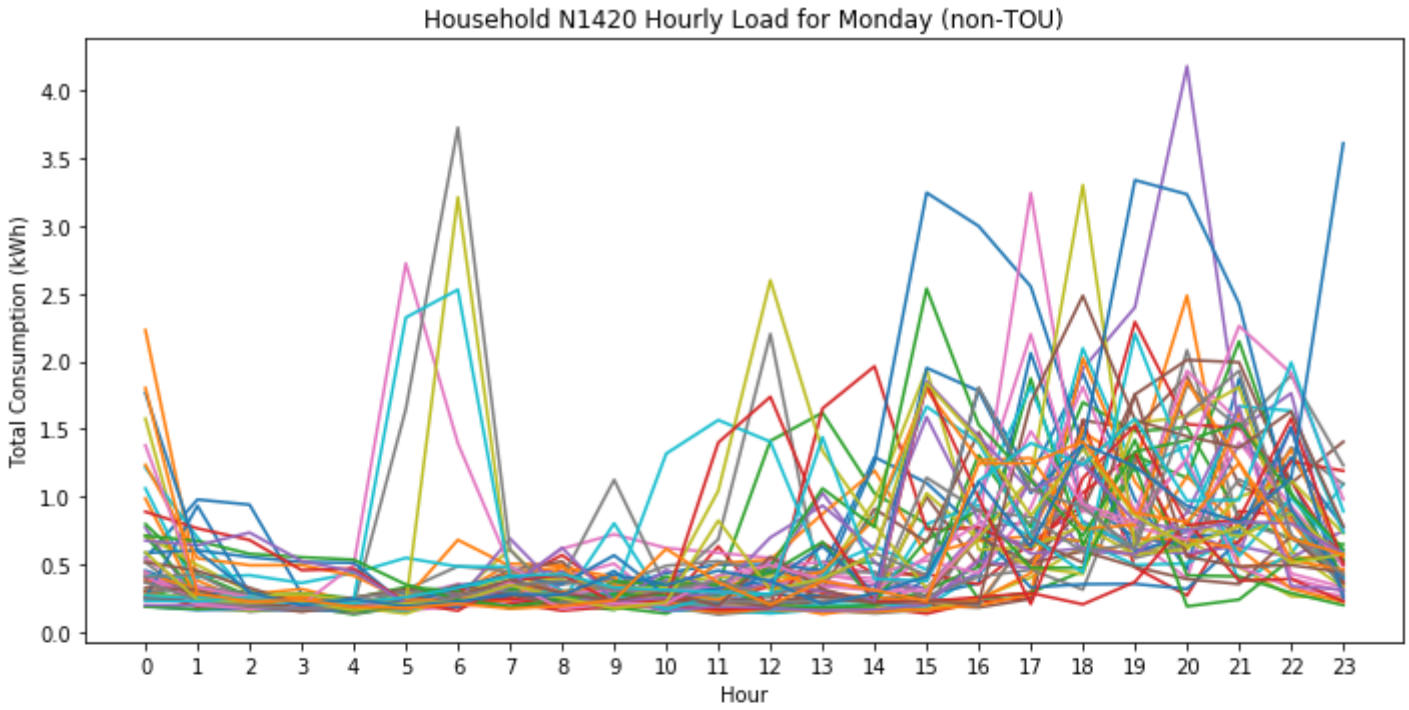

Time: 23:00
TOU Default Price
non-TOU Estimation
Consumption (kWh)
0.55
0.50
0.45
0.40

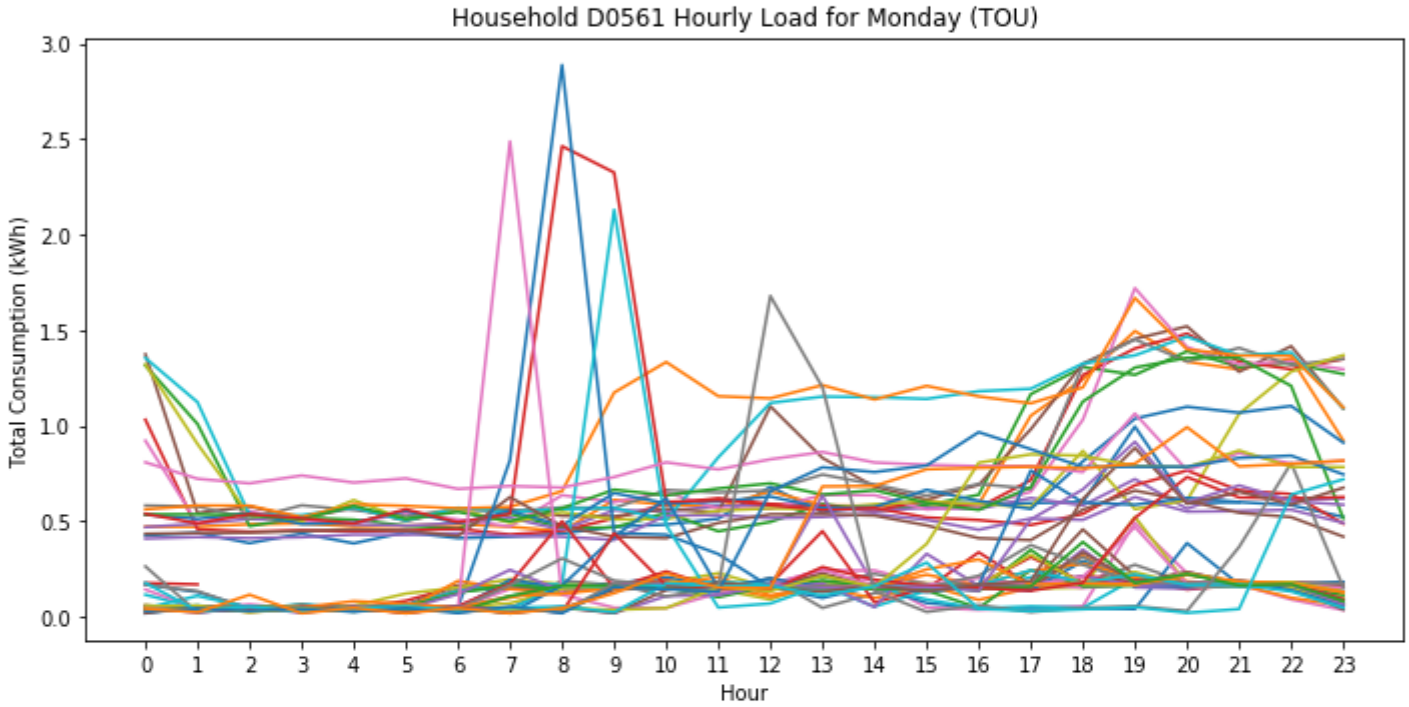


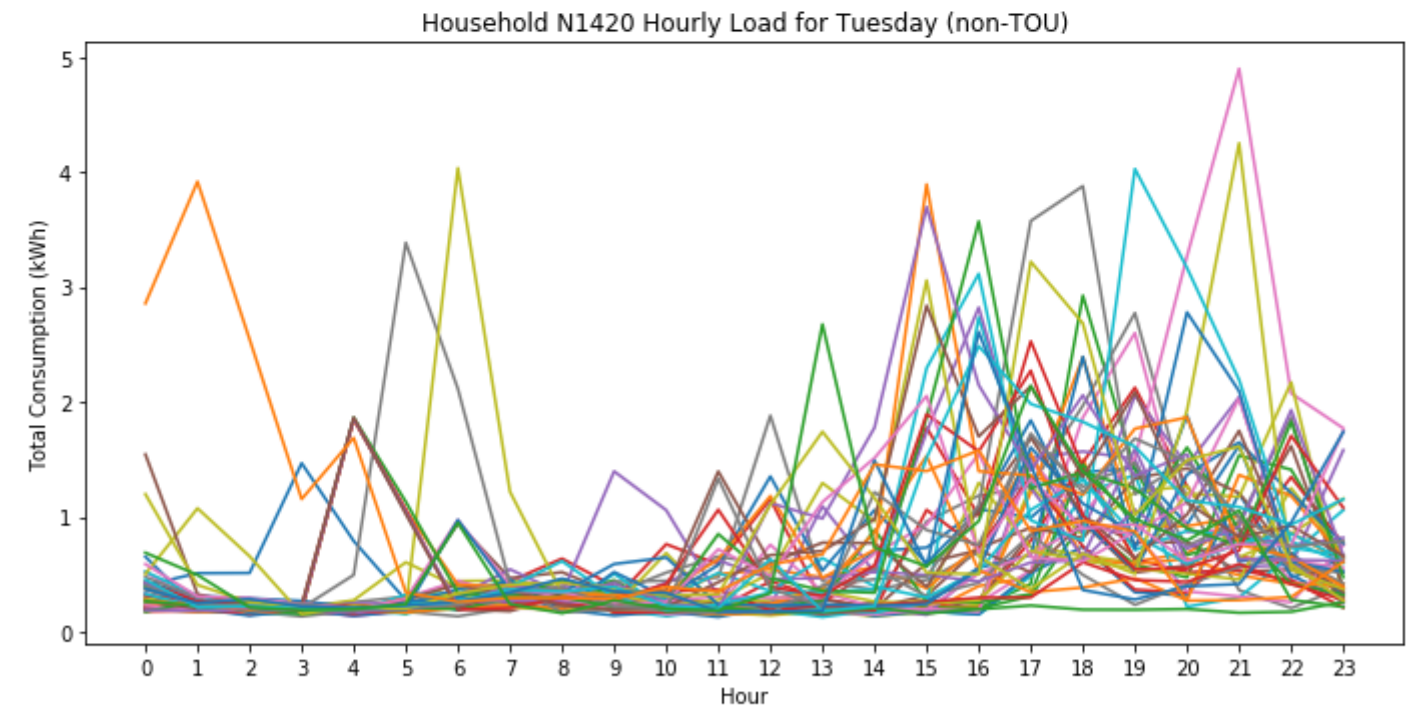
The above figures somehow shows that the estimation of TOU group facing default price using non-TOU data (removing the const difference) is close to the real TOU groupd facing default price.

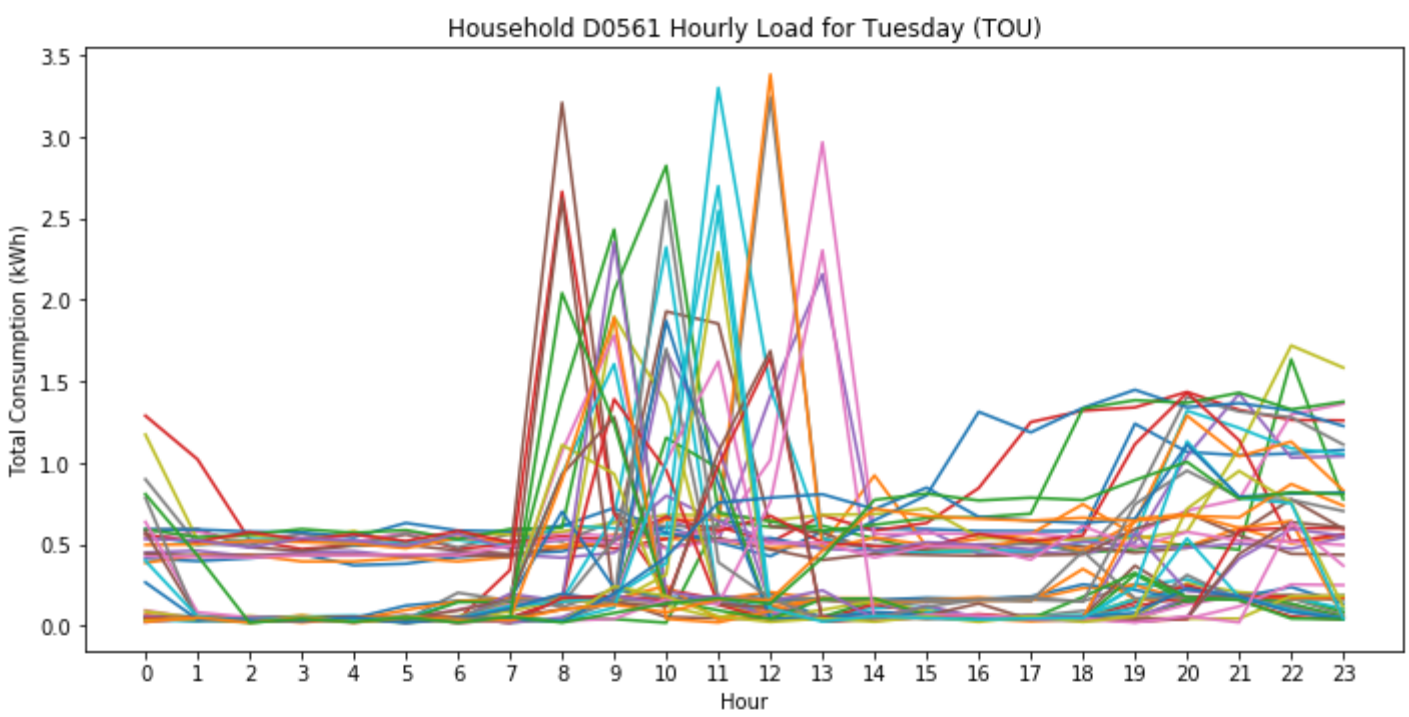
Study the household consumption types. We first take a look at how does the demand look like in differnt work days over a year

In [146]:
```
# We just randomly pick one non-TOU household and one TOU houdsehold
import random
id_Ntou = random.randint(0, df_Ntou1h.shape[1] - 3) #make sure number is in household index
id_tou = random.randint(0, df_tou1h.shape[1] - 3)
columns_Ntou = df_Ntou1h.columns.values[id_Ntou + 1] #name of the household
columns_tou = df_tou1h.columns.values[id_tou + 1]
# select all Monday data
week_days = [1, 2, 3, 4, 5, 6, 7]
label_week = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
fig_all = plt.figure(figsize = (10, 70))
ax = [] # store subplot objects

for j in range(len(week_days)):
    ax.append(fig_all.add_subplot(14, 1, 2 * j + 1))
    df_Ntou1h_Day = df_Ntou1h[pd.to_datetime(df_tariff_1h.GMT).dt.weekday == j]
    for i in range(int(df_Ntou1h_Day.shape[0] / 24)):
        ax[-1].plot(range(24), df_Ntou1h_Day[columns_Ntou].iloc[i * 24 : i * 24 + 24], label = label_week[j])
    ax[-1].set_title('Household ' + columns_Ntou + ' Hourly Load for ' + label_week[j]+ ' (non-TOU)')
    ax[-1].set_xlabel('Hour')
    ax[-1].set_xticks(range(24))
    ax[-1].set_ylabel('Total Consumption (kWh)')


for j in range(len(week_days)):
    ax.append(fig_all.add_subplot(14, 1, 2 * j + 2))
    df_tou1h_Day = df_tou1h[pd.to_datetime(df_tariff_1h.GMT).dt.weekday == j]
    for i in range(int(df_tou1h_Day.shape[0] / 24)):
        ax[-1].plot(range(24), df_tou1h_Day[columns_tou].iloc[i * 24 : i * 24 + 24], label = label_week[j])
    ax[-1].set_title('Household ' + columns_tou + ' Hourly Load for ' + label_week[j]+ ' (TOU)')
    ax[-1].set_xlabel('Hour')
    ax[-1].set_xticks(range(24))
    ax[-1].set_ylabel('Total Consumption (kWh)')
plt.tight_layout()
```

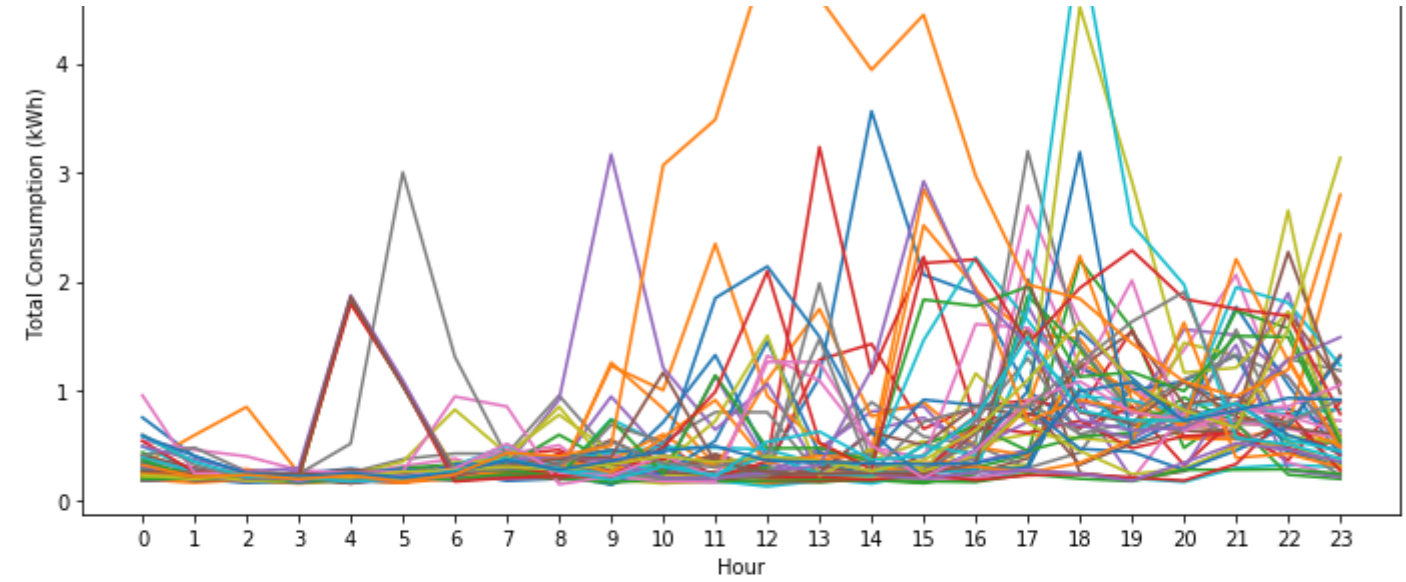

Household N1420 Hourly Load for Monday (non-TOU)
Total Consumption (kWh)
Hour

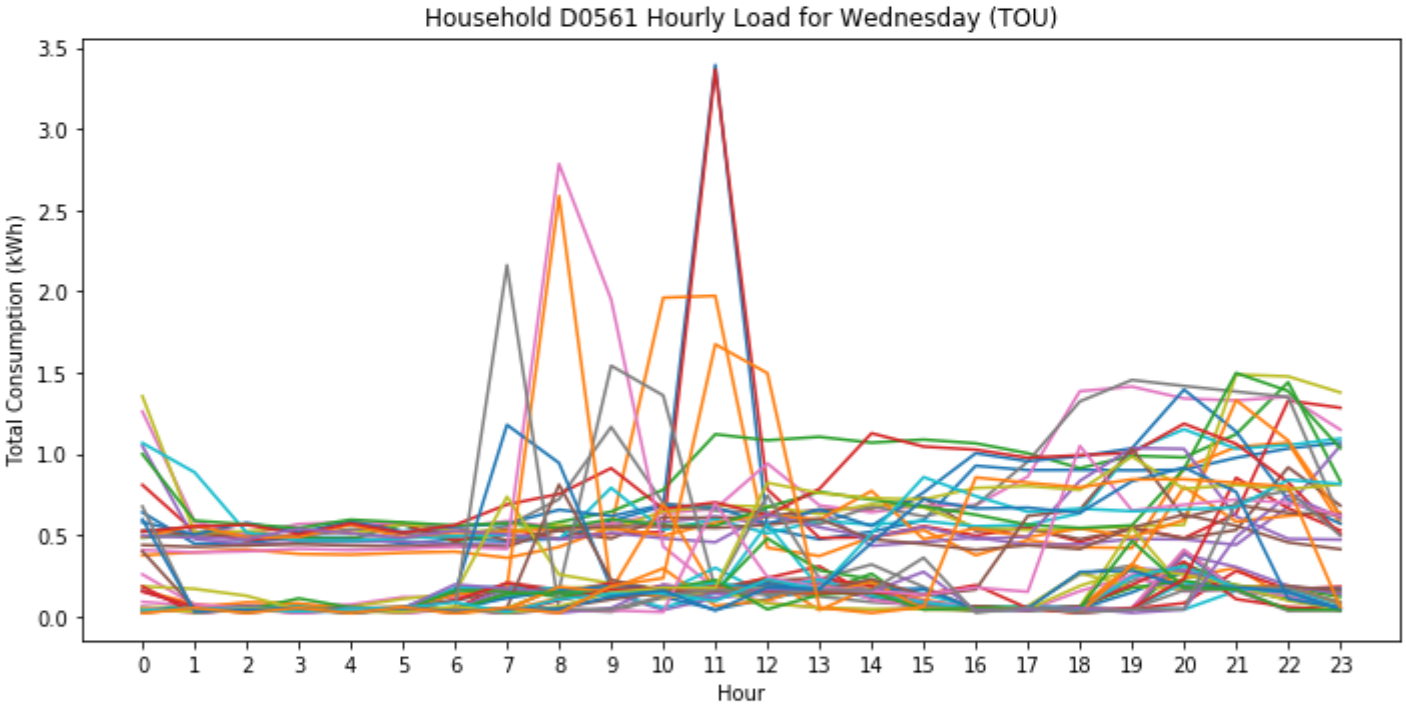

Household D0561 Hourly Load for Monday (TOU)
Total Consumption (kWh)
Hour

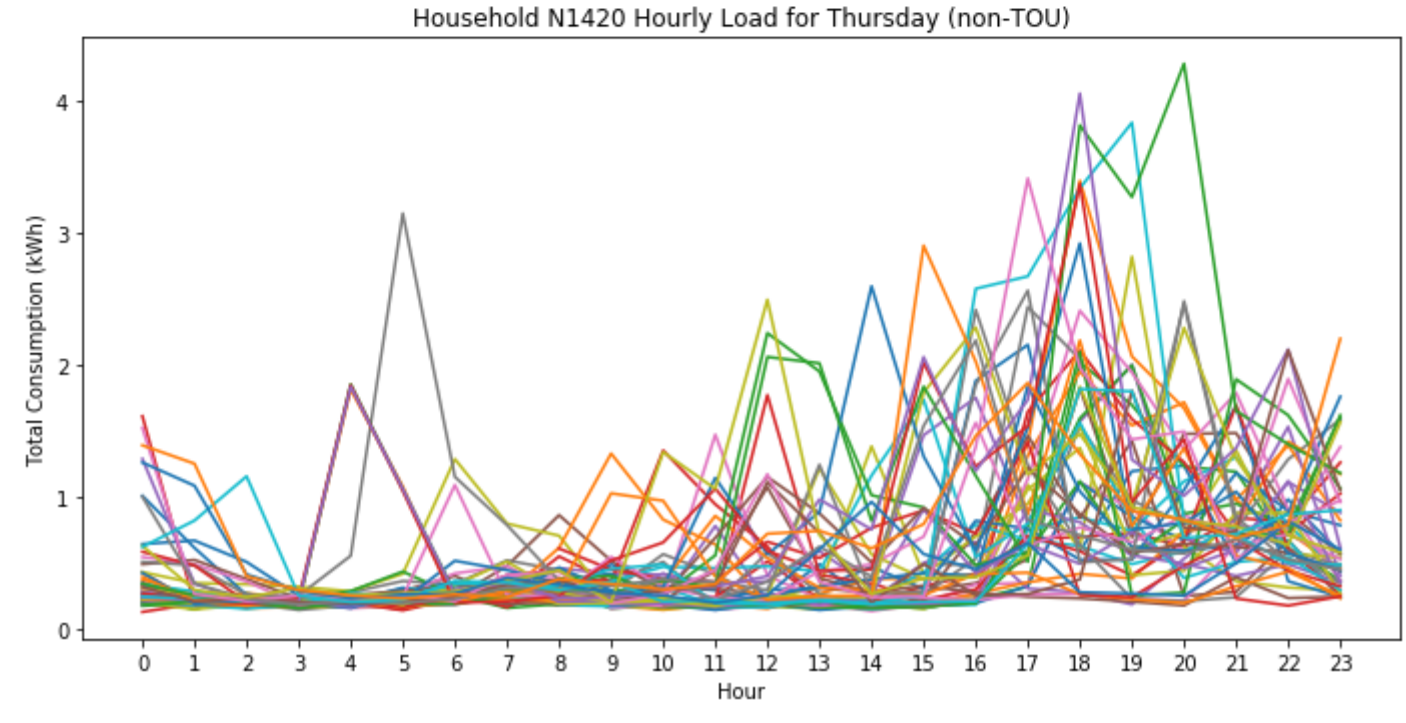

Household N1420 Hourly Load for Tuesday (non-TOU)
Total Consumption (kWh)
Hour

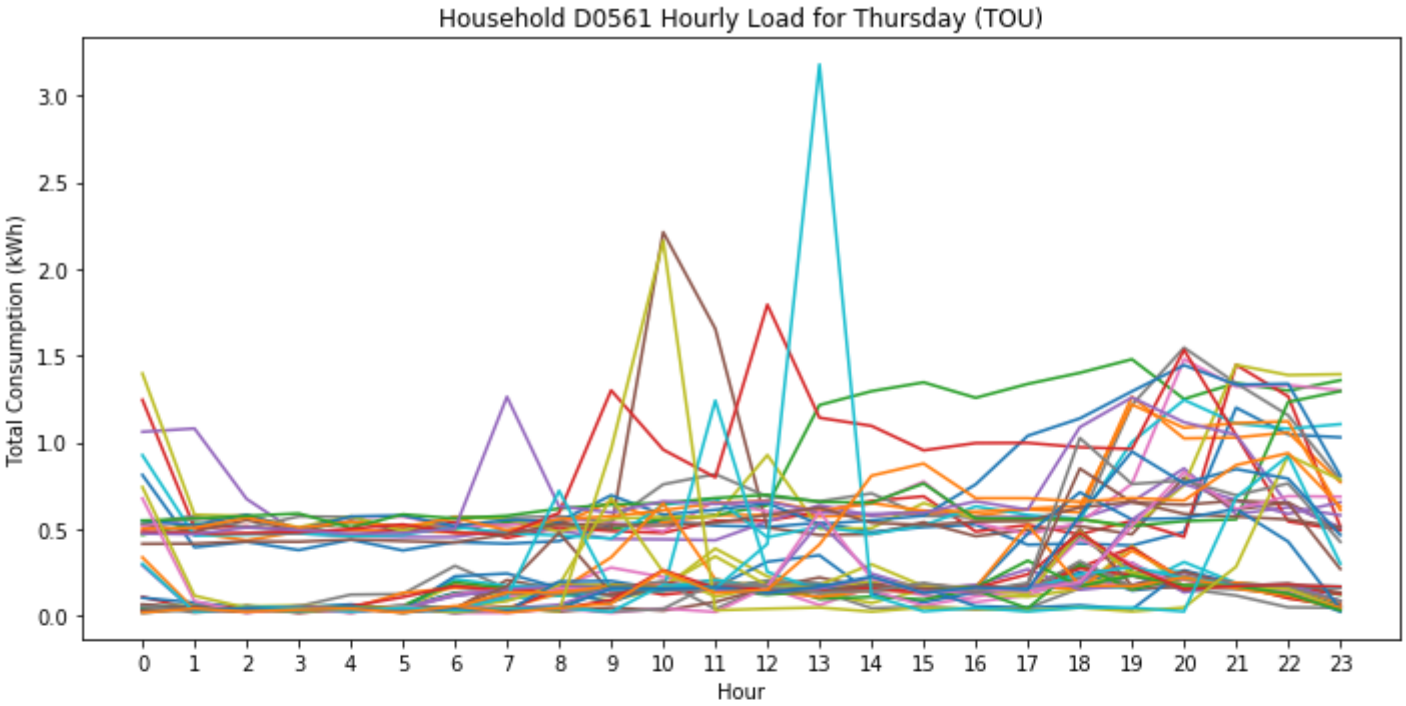

Household D0561 Hourly Load for Tuesday (TOU)
Total Consumption (kWh)
Hour

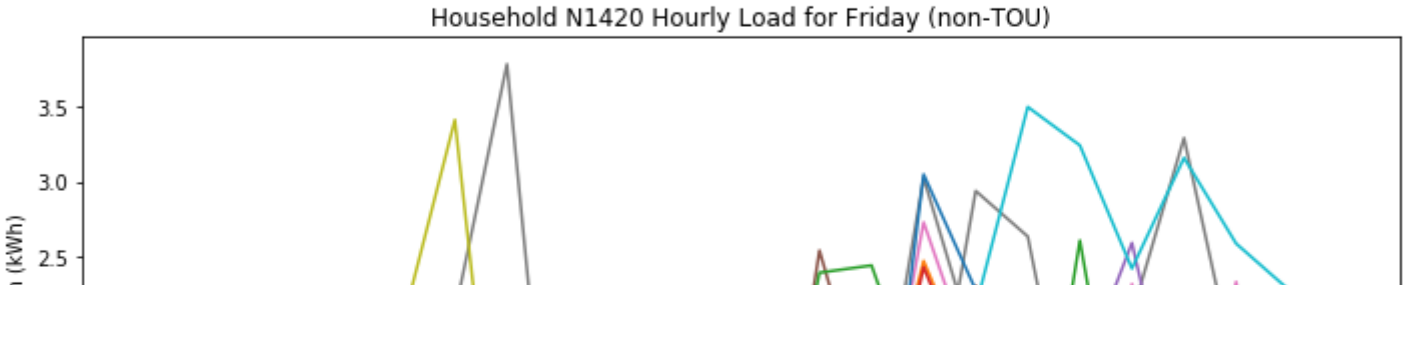
Household N1420 Hourly Load for Wednesday (non-TOU)

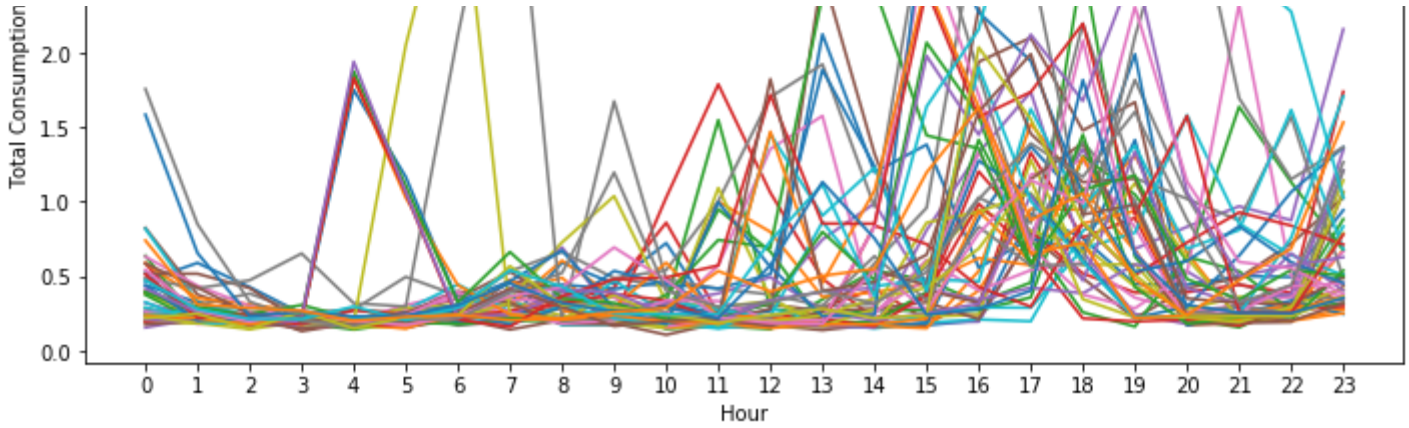

Total Consumption (kWh)
Hour

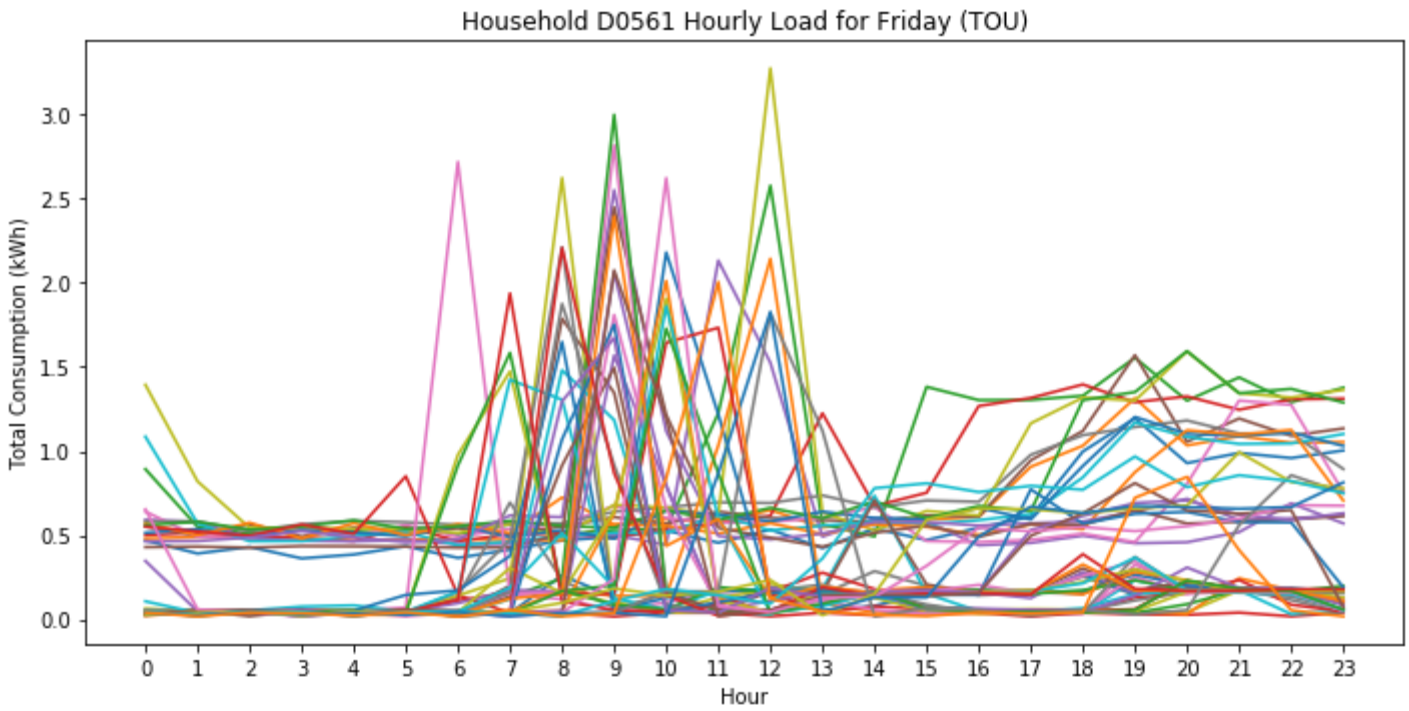

Household D0561 Hourly Load for Wednesday (TOU)
Total Consumption (kWh)
Hour

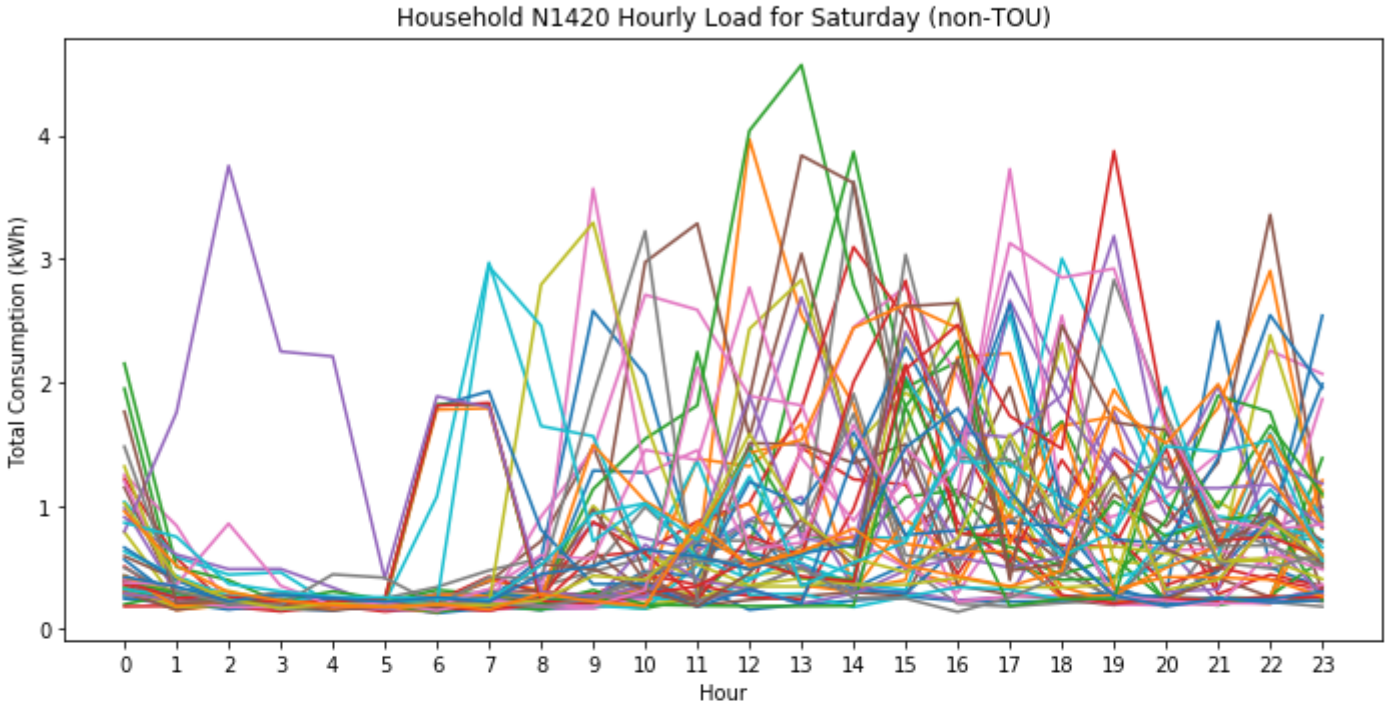

Household N1420 Hourly Load for Thursday (non-TOU)
Total Consumption (kWh)
Hour

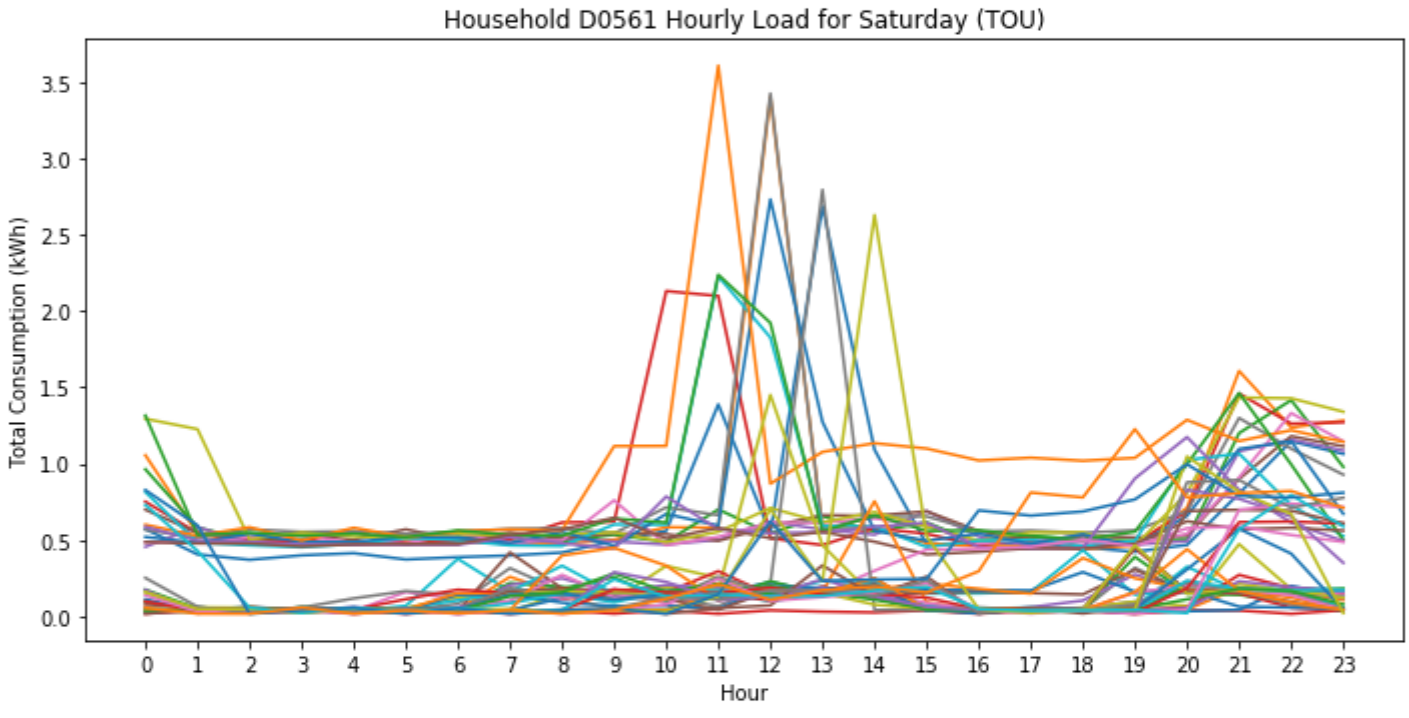

Household D0561 Hourly Load for Thursday (TOU)
Total Consumption (kWh)
Hour


Household N1420 Hourly Load for Friday (non-TOU)

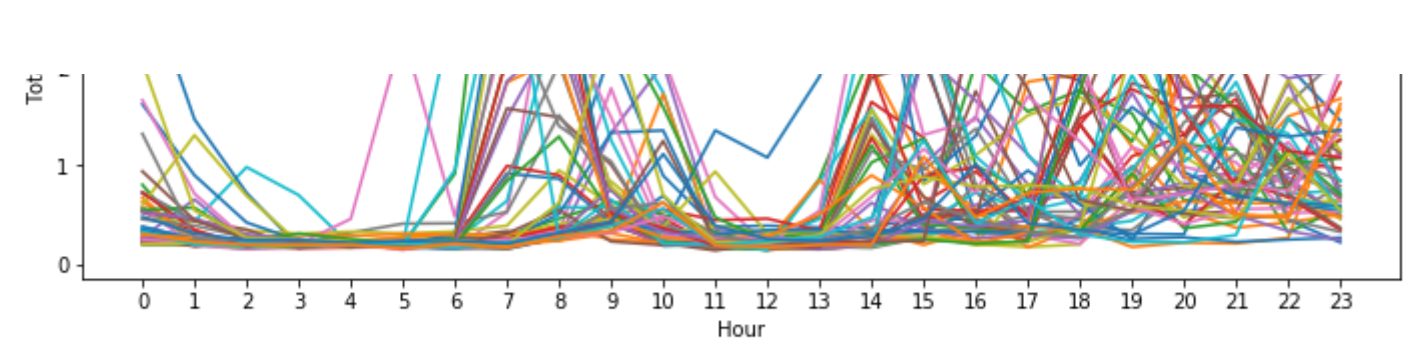

Total Consumption
0.0
0.5
1.0
1.5
2.0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

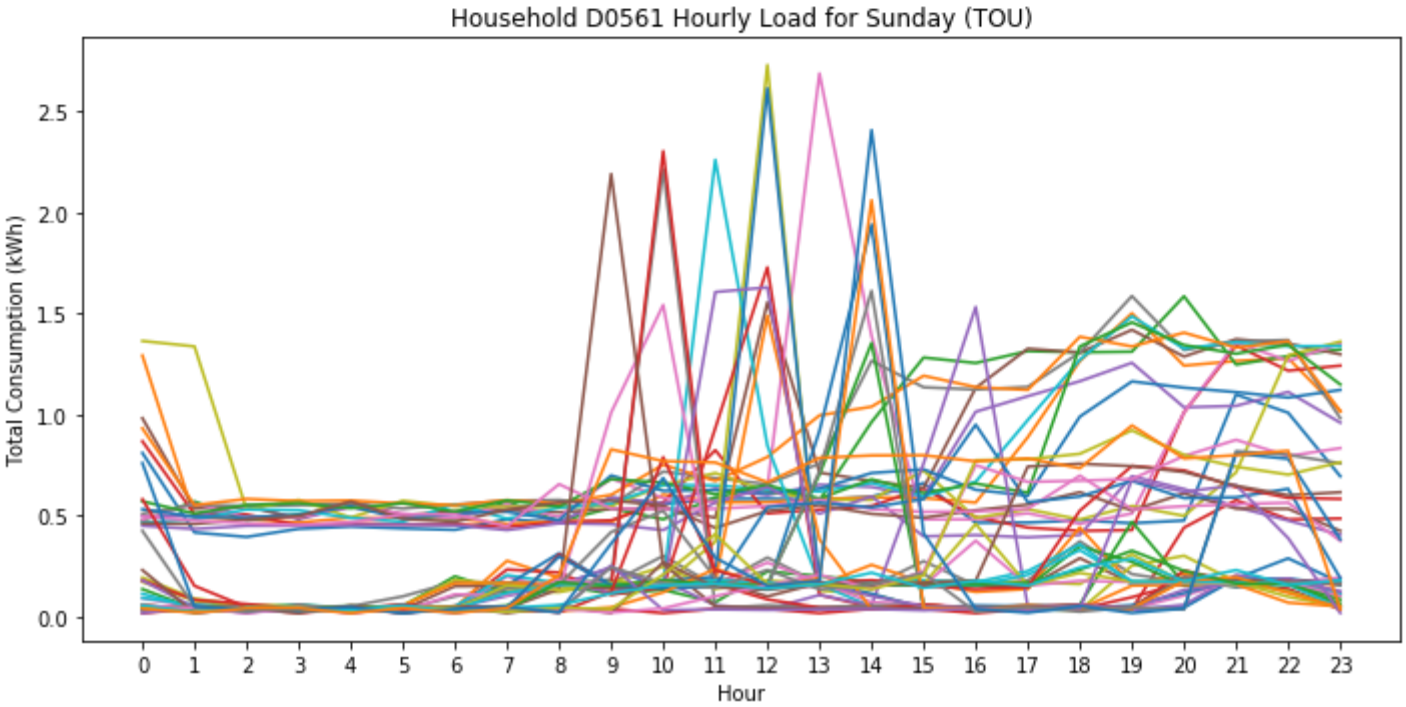

Household D0561 Hourly Load for Friday (TOU)
Total Consumption (kWh)
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour


Household N1420 Hourly Load for Saturday (non-TOU)
Total Consumption (kWh)
0
1
2
3
4
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour


Household D0561 Hourly Load for Saturday (TOU)
Total Consumption (kWh)
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

Household N1420 Hourly Load for Sunday (non-TOU)
al Consumption (kWh)
2
3
4
5





In [133]:
```
df_Ntou1h_Day.shape
```

Out[133]:
```
(1248, 4177)
```

In [ ]:
```
import math
math.isnan(float('nan'))
# float('nan')
```

In [ ]:
```
temp_df_tariff[:-11]
```

In [ ]:
```
# Now we compress the rows such that the frequency is 1 hour
# first let's delete the GMT index of matrix consists in odd rows
odd_df_ntou = df_ntou_2013.loc[1::2].copy()
odd_df_ntou.loc[:, 'GMT'] = ''
# create the new dataframe that is the sum of odd and even rows
df_ntou_2013_1h = (df_ntou_2013[::2] + odd_df_ntou.values).reset_index(drop = True)
# store it to a csv file
df_ntou_2013_1h.to_csv("C:\\Users\\Rockwell\\Desktop\\Paper4\\data_collection\\data_tables\\Consumption_Ntou2013_1h.csv", index = False)
```

Study the consumer types, lifestyle for 6 am case and the seasonal effect and also .

Next, we can add one more filtering condition, check the hourly relation in a specific day of a week over the whole year.

Scaterplot the consumption-weather relationship for the non-TOU case

In [ ]:
```
x = np.arange(0, 2, 0.1)
# plt.plot(x, x, label = 'linear')
# plt.plot(x, x**2, label = 'quadratic')
fig, ax = plt.subplots(1,2)
ax[1].plot(x, x, 'ro',label = 'linear')
ax[1].legend()
ax[0].plot(x, x**2, label = 'quatratic')
ax[0].legend()
ax[0].xlabel('X axis')
ax[0].ylabel('Y axis')
plt.show()
```

In [ ]:
```
# We first right join the df_wea1h to df_Ntou1h.
df_touWea1h = pd.merge(df_Ntou1h, df_wea1h, on = 'GMT', how = 'right')

ax1 = df_touWea1h.plot.scatter(x = 'TempC', y = 'N0000', c = 'Blue')
```

(Just a reminder, for Matplotlib the pandas data object may or may not work as intended, so it's better to convert array to np.array objects or np.ma.maksed_array objects as inputs.)