In [1]:
```
!pip install plotly
```

```
Requirement already satisfied: plotly in c:\users\arun kalaeswaran\anaconda3\lib\site-packages (4.5.
0)
Requirement already satisfied: six in c:\users\arun kalaeswaran\anaconda3\lib\site-packages (from pl
otly) (1.12.0)
Requirement already satisfied: retrying>=1.3.3 in c:\users\arun kalaeswaran\anaconda3\lib\site-packa
ges (from plotly) (1.3.3)
```

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.simplefilter('ignore')
import gc
%matplotlib inline
from plotly import tools, subplots
import plotly.offline as py
py.init_notebook_mode(connected=True)
import plotly.graph_objs as go
import plotly.express as px
pd.set_option('max_columns', 100)
```

Loading Datasets

In [2]:
```
building_data = pd.read_csv('building_metadata.csv')
weather_train = pd.read_csv('weather_train.csv')
weather_test = pd.read_csv('weather_test.csv')
train = pd.read_csv('train.csv')
test = pd.read_csv('test.csv')
```

In [3]:
```
## Function to reduce the DF size
def reduce_mem_usage(df, verbose=True):
    numerics = ['int16', 'int32', 'int64', 'float16', 'float32', 'float64']
    start_mem = df.memory_usage().sum() / 1024**2
    for col in df.columns:
        col_type = df[col].dtypes
        if col_type in numerics:
            c_min = df[col].min()
            c_max = df[col].max()
            if str(col_type)[:3] == 'int':
                if c_min > np.iinfo(np.int8).min and c_max < np.iinfo(np.int8).max:
                    df[col] = df[col].astype(np.int8)
                elif c_min > np.iinfo(np.int16).min and c_max < np.iinfo(np.int16).max:
                    df[col] = df[col].astype(np.int16)
                elif c_min > np.iinfo(np.int32).min and c_max < np.iinfo(np.int32).max:
                    df[col] = df[col].astype(np.int32)
                elif c_min > np.iinfo(np.int64).min and c_max < np.iinfo(np.int64).max:
                    df[col] = df[col].astype(np.int64)
            else:
                if c_min > np.finfo(np.float16).min and c_max < np.finfo(np.float16).max:
                    df[col] = df[col].astype(np.float16)
                elif c_min > np.finfo(np.float32).min and c_max < np.finfo(np.float32).max:
                    df[col] = df[col].astype(np.float32)
                else:
                    df[col] = df[col].astype(np.float64)
    end_mem = df.memory_usage().sum() / 1024**2
    if verbose: print('Mem. usage decreased to {:5.2f} Mb ({:.1f}% reduction)'.format(end_mem, 100 *
(start_mem - end_mem) / start_mem))
    return df
```

In [4]:
```
## REducing memory
train = reduce_mem_usage(train)
test = reduce_mem_usage(test)

weather_train = reduce_mem_usage(weather_train)
weather_test = reduce_mem_usage(weather_test)
building_data = reduce_mem_usage(building_data)
```

```
Mem. usage decreased to 289.19 Mb (53.1% reduction)
Mem. usage decreased to 596.49 Mb (53.1% reduction)
Mem. usage decreased to  3.07 Mb (68.1% reduction)
Mem. usage decreased to  6.08 Mb (68.1% reduction)
Mem. usage decreased to  0.03 Mb (60.3% reduction)
```

Count and info of rows and Columns in dataset

In [6]:
```
print(building_data.head(5))
# print(building_data.tail(5))
print(building_data.info())
```

```
   site_id  building_id primary_use  square_feet  year_built  floor_count
0        0            0   Education         7432      2008.0          NaN
1        0            1   Education         2720      2004.0          NaN
2        0            2   Education         5376      1991.0          NaN
3        0            3   Education        23685      2002.0          NaN
4        0            4   Education       116607      1975.0          NaN
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1449 entries, 0 to 1448
Data columns (total 6 columns):
site_id        1449 non-null int8
building_id    1449 non-null int16
primary_use    1449 non-null object
square_feet    1449 non-null int32
year_built     675 non-null float16
floor_count    355 non-null float16
dtypes: float16(2), int16(1), int32(1), int8(1), object(1)
memory usage: 27.0+ KB
None
```

In [7]:
```
print(weather_train.head(5))
# print(weather_train.tail(5))
print(weather_train.info())
```

```
   site_id            timestamp  air_temperature  cloud_coverage  \
0        0  2016-01-01 00:00:00        25.000000             6.0
1        0  2016-01-01 01:00:00        24.406250             NaN
2        0  2016-01-01 02:00:00        22.796875             2.0
3        0  2016-01-01 03:00:00        21.093750             2.0
4        0  2016-01-01 04:00:00        20.000000             2.0

   dew_temperature  precip_depth_1_hr  sea_level_pressure  wind_direction  \
0         20.00000                NaN              1019.5             0.0
1         21.09375               -1.0              1020.0            70.0
2         21.09375                0.0              1020.0             0.0
3         20.59375                0.0              1020.0             0.0
4         20.00000               -1.0              1020.0           250.0

   wind_speed
0    0.000000
1    1.500000
2    0.000000
3    0.000000
4    2.599609
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 139773 entries, 0 to 139772
Data columns (total 9 columns):
site_id               139773 non-null int8
timestamp             139773 non-null object
air_temperature       139718 non-null float16
cloud_coverage        70600 non-null float16
dew_temperature       139660 non-null float16
precip_depth_1_hr     89484 non-null float16
sea_level_pressure    129155 non-null float16
wind_direction        133505 non-null float16
wind_speed            139469 non-null float16
dtypes: float16(7), int8(1), object(1)
memory usage: 3.1+ MB
None
```

In [8]:
```
print(weather_test.head(5))
# print(weather_test.tail(5))
print(weather_test.info())
```

```
   site_id            timestamp  air_temperature  cloud_coverage  \
0        0  2017-01-01 00:00:00        17.796875             4.0
1        0  2017-01-01 01:00:00        17.796875             2.0
2        0  2017-01-01 02:00:00        16.093750             0.0
3        0  2017-01-01 03:00:00        17.203125             0.0
4        0  2017-01-01 04:00:00        16.703125             2.0

   dew_temperature  precip_depth_1_hr  sea_level_pressure  wind_direction  \
0        11.703125                NaN              1021.5           100.0
1        12.796875                0.0              1022.0           130.0
2        12.796875                0.0              1022.0           140.0
3        13.296875                0.0              1022.0           140.0
4        13.296875                0.0              1022.5           130.0

   wind_speed
0    3.599609
1    3.099609
2    3.099609
3    3.099609
4    2.599609
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 277243 entries, 0 to 277242
Data columns (total 9 columns):
site_id               277243 non-null int8
timestamp             277243 non-null object
air_temperature       277139 non-null float16
cloud_coverage        136795 non-null float16
dew_temperature       276916 non-null float16
precip_depth_1_hr     181655 non-null float16
sea_level_pressure    255978 non-null float16
wind_direction        264873 non-null float16
wind_speed            276783 non-null float16
dtypes: float16(7), int8(1), object(1)
memory usage: 6.1+ MB
None
```

Train and Test description

In [9]:
```
train.describe(include='all')
```

Out[9]:

	building_id	meter	timestamp	meter_reading
count	2.021610e+07	2.021610e+07	20216100	2.021610e+07
unique	NaN	NaN	8784	NaN
top	NaN	NaN	2016-12-27 22:00:00	NaN
freq	NaN	NaN	2370	NaN
mean	7.992780e+02	6.624412e-01	NaN	1.988706e+03
std	4.269133e+02	9.309921e-01	NaN	1.532159e+05
min	0.000000e+00	0.000000e+00	NaN	0.000000e+00
25%	3.930000e+02	0.000000e+00	NaN	1.830000e+01
50%	8.950000e+02	0.000000e+00	NaN	7.877500e+01
75%	1.179000e+03	1.000000e+00	NaN	2.679840e+02
max	1.448000e+03	3.000000e+00	NaN	2.190470e+07

In [10]:
```
test.describe(include='all')
```

Out[10]:

	row_id	building_id	meter	timestamp
count	4.169760e+07	4.169760e+07	4.169760e+07	41697600
unique	NaN	NaN	NaN	17520
top	NaN	NaN	NaN	2018-12-10 12:00:00
freq	NaN	NaN	NaN	2380
mean	2.084880e+07	8.075824e+02	6.642857e-01	NaN
std	1.203706e+07	4.297680e+02	9.278067e-01	NaN
min	0.000000e+00	0.000000e+00	0.000000e+00	NaN
25%	1.042440e+07	4.047500e+02	0.000000e+00	NaN
50%	2.084880e+07	9.000000e+02	0.000000e+00	NaN
75%	3.127320e+07	1.194250e+03	1.000000e+00	NaN
max	4.169760e+07	1.448000e+03	3.000000e+00	NaN

Merging building and weather data into test and train

In [5]:
```
train = train.merge(building_data, on='building_id', how='left')
test = test.merge(building_data, on='building_id', how='left')

train = train.merge(weather_train, on=['site_id', 'timestamp'], how='left')
test = test.merge(weather_test, on=['site_id', 'timestamp'], how='left')
```

Count of Data Missing in Train

In [12]:
```
for col in train.columns:
    if train[col].isna().sum()>0:
        print (col,train[col].isna().sum())
```

```
year_built 12127645
floor_count 16709167
air_temperature 96658
cloud_coverage 8825365
dew_temperature 100140
precip_depth_1_hr 3749023
sea_level_pressure 1231669
wind_direction 1449048
wind_speed 143676
```

Count of Data Missing in test

In [13]:
```
for col in test.columns:
    if test[col].isna().sum()>0:
        print (col,test[col].isna().sum())
```

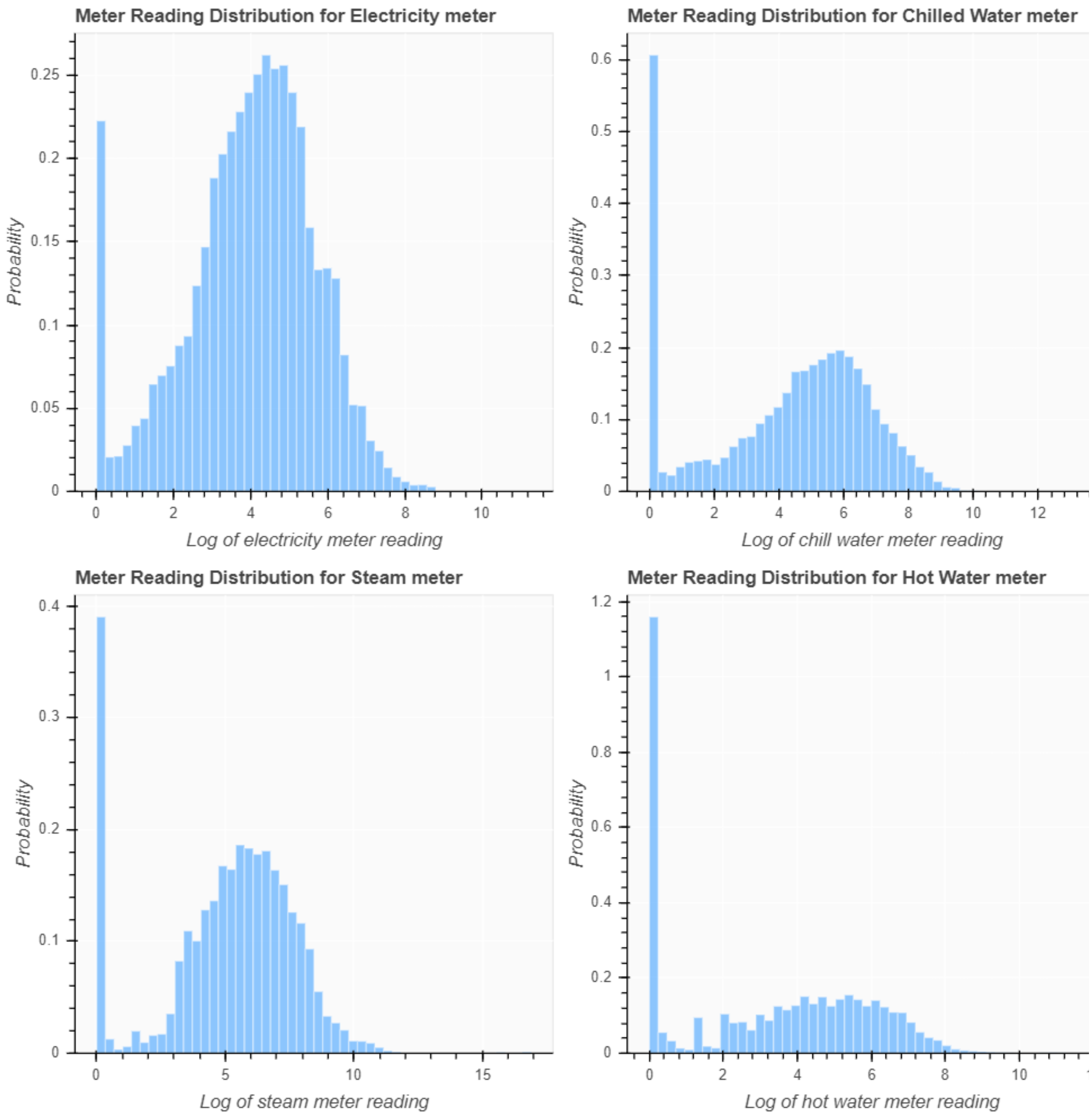
```
year_built 24598080
floor_count 34444320
air_temperature 221901
cloud_coverage 19542180
dew_temperature 260799
precip_depth_1_hr 7801563
sea_level_pressure 2516826
wind_direction 2978663
wind_speed 302089
```

Distribution plot for the target variable

In [14]:
```
from bokeh.layouts import gridplot
from bokeh.plotting import figure, show, output_file
from bokeh.io import output_notebook

output_notebook()

def make_plot(title, hist, edges, xlabel):
    p = figure(title=title, tools='', background_fill_color="#fafafa")
    p.quad(top=hist, bottom=0, left=edges[:-1], right=edges[1:],
           fill_color="#1E90FF", line_color="white", alpha=0.5)
    p.y_range.start = 0
    p.xaxis.axis_label = f'Log of {xlabel} meter reading'
    p.yaxis.axis_label = 'Probability'
    p.grid.grid_line_color="white"
    return p

def scatter_plot(cnt_srs, color):
    trace = go.Scatter(
        x=cnt_srs.index[::-1],
        y=cnt_srs.values[::-1],
        showlegend=False,
        marker=dict(
            color=color,
        ),
    )
    return trace
```

(https://bokeh.pydata.org) BokehJS 1.2.0 successfully loaded.

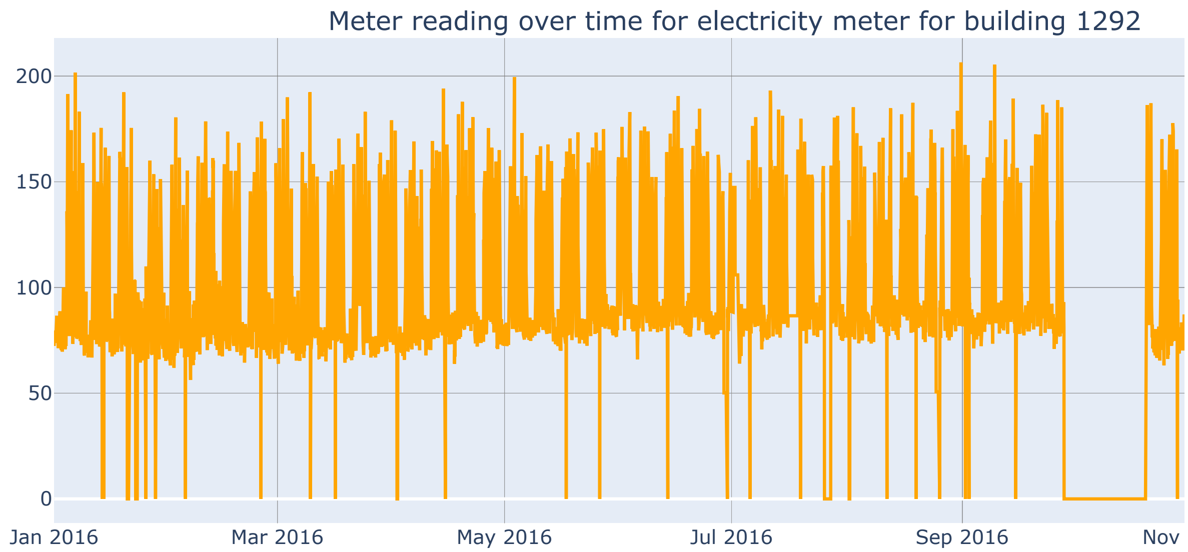
In [15]:
```
dataset = train[train["meter"]==0]
hist, edges = np.histogram(np.log1p(dataset["meter_reading"].values), density=True, bins=50)
p1 = make_plot("Meter Reading Distribution for Electricity meter", hist, edges, "electricity")

dataset = train[train["meter"]==1]
hist, edges = np.histogram(np.log1p(dataset["meter_reading"].values), density=True, bins=50)
p2 = make_plot("Meter Reading Distribution for Chilled Water meter", hist, edges, 'chill water')

dataset = train[train["meter"]==2]
hist, edges = np.histogram(np.log1p(dataset["meter_reading"].values), density=True, bins=50)
p3 = make_plot("Meter Reading Distribution for Steam meter", hist, edges, 'steam')

dataset = train[train["meter"]==3]
hist, edges = np.histogram(np.log1p(dataset["meter_reading"].values), density=True, bins=50)
p4 = make_plot("Meter Reading Distribution for Hot Water meter", hist, edges, 'hot water')

show(gridplot([p1,p2,p3,p4], ncols=2, plot_width=400, plot_height=400, toolbar_location=None))
```

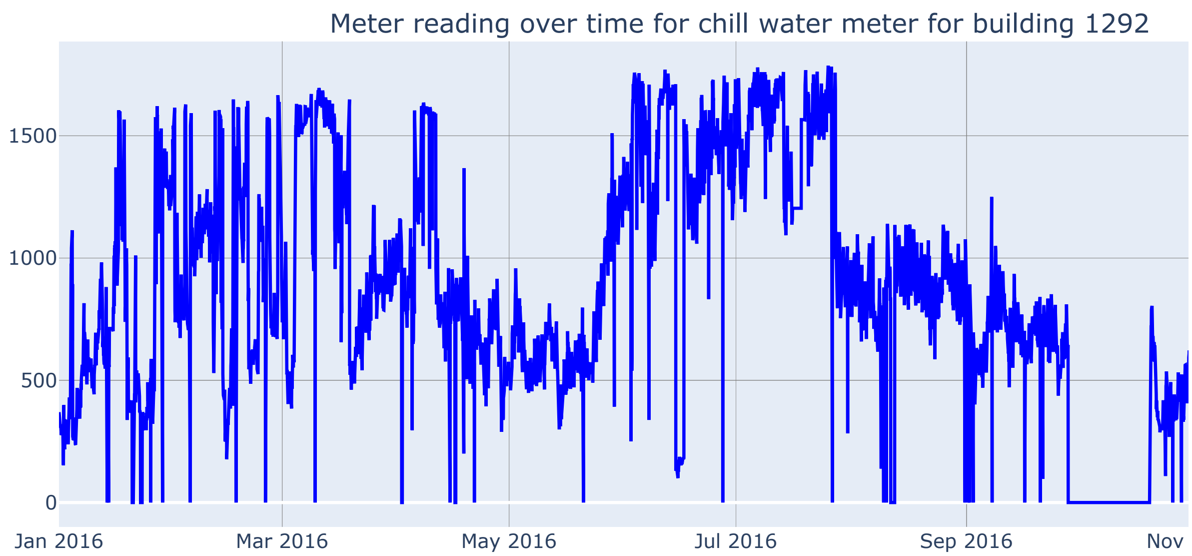


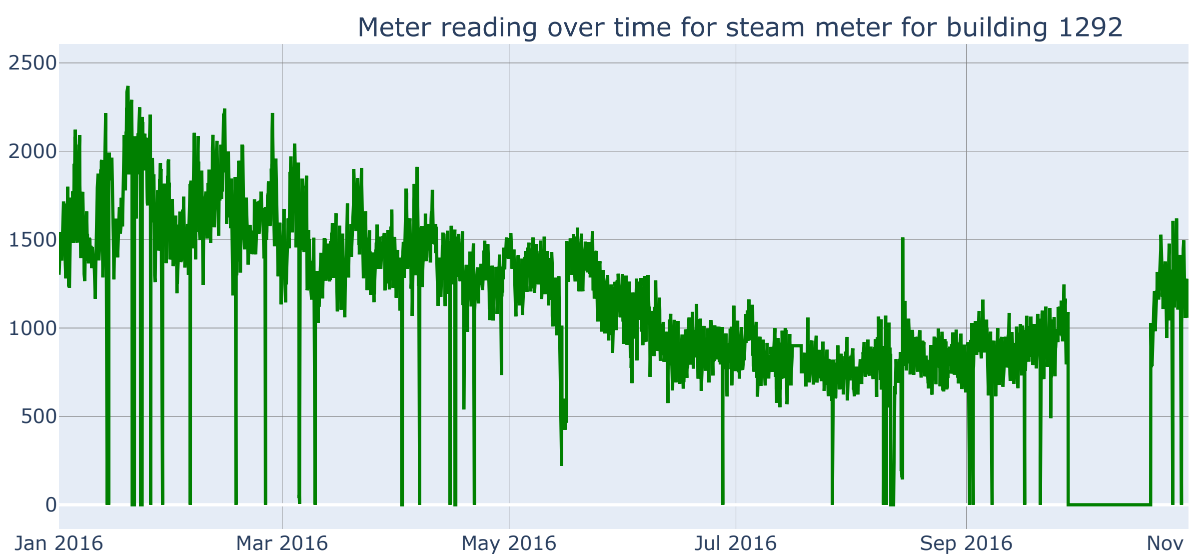
Electricty meter type has the most number of rows

Distribution of target variable meter readings over a time period

In [16]:
```
dataset = train[train["building_id"]==1292].reset_index(drop=True)


tdf = dataset[dataset["meter"]==0]
col = tdf["meter_reading"]
col.index = tdf["timestamp"]
trace1 = scatter_plot(col, 'orange')

tdf = dataset[dataset["meter"]==1]
col = tdf["meter_reading"]
col.index = tdf["timestamp"]
trace2 = scatter_plot(col, 'blue')

tdf = dataset[dataset["meter"]==2]
col = tdf["meter_reading"]
col.index = tdf["timestamp"]
trace3 = scatter_plot(col, 'green')


subtitles = ["Meter reading over time for electricity meter for building 1292",
             "Meter reading over time for chill water meter for building 1292",
             "Meter reading over time for steam meter for building 1292",
            ]
fig = subplots.make_subplots(rows=3, cols=1, vertical_spacing=0.06,
                          subplot_titles=subtitles)
fig.append_trace(trace1, 1, 1)
fig.append_trace(trace2, 2, 1)
fig.append_trace(trace3, 3, 1)
fig['layout'].update(height=1200, width=1000, paper_bgcolor='rgb(223,223,233)')
py.iplot(fig, filename='meter-plots')
```


Meter reading over time for electricity meter for building 1292
200
150
100
50
0
Jan 2016
Mar 2016
May 2016
Jul 2016
Sep 2016
Nov


Meter reading over time for chill water meter for building 1292
1500
1000
500
0
Jan 2016
Mar 2016
May 2016
Jul 2016
Sep 2016
Nov


Meter reading over time for steam meter for building 1292
2500
2000
1500
1000
500
0
Jan 2016
Mar 2016
May 2016
Jul 2016
Sep 2016
Nov

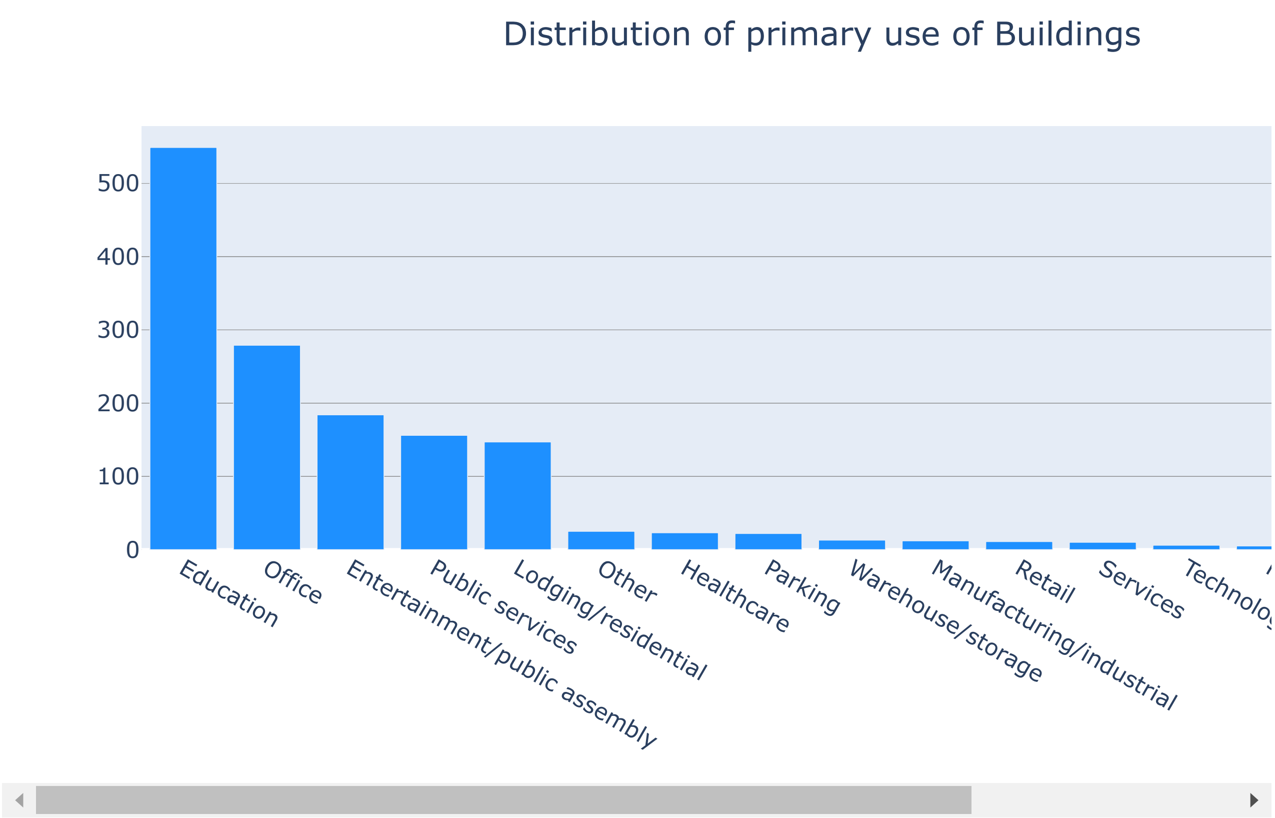
The electricity meter readings are generally in the range of 60 to 400 but becomes 0 at times in between.

We can see an increase in the chill water meter from sep to octof 2016 for this building probably due to summer time

Distribution plots for Building

Energy consumption in building will if it is used for commerical purpose

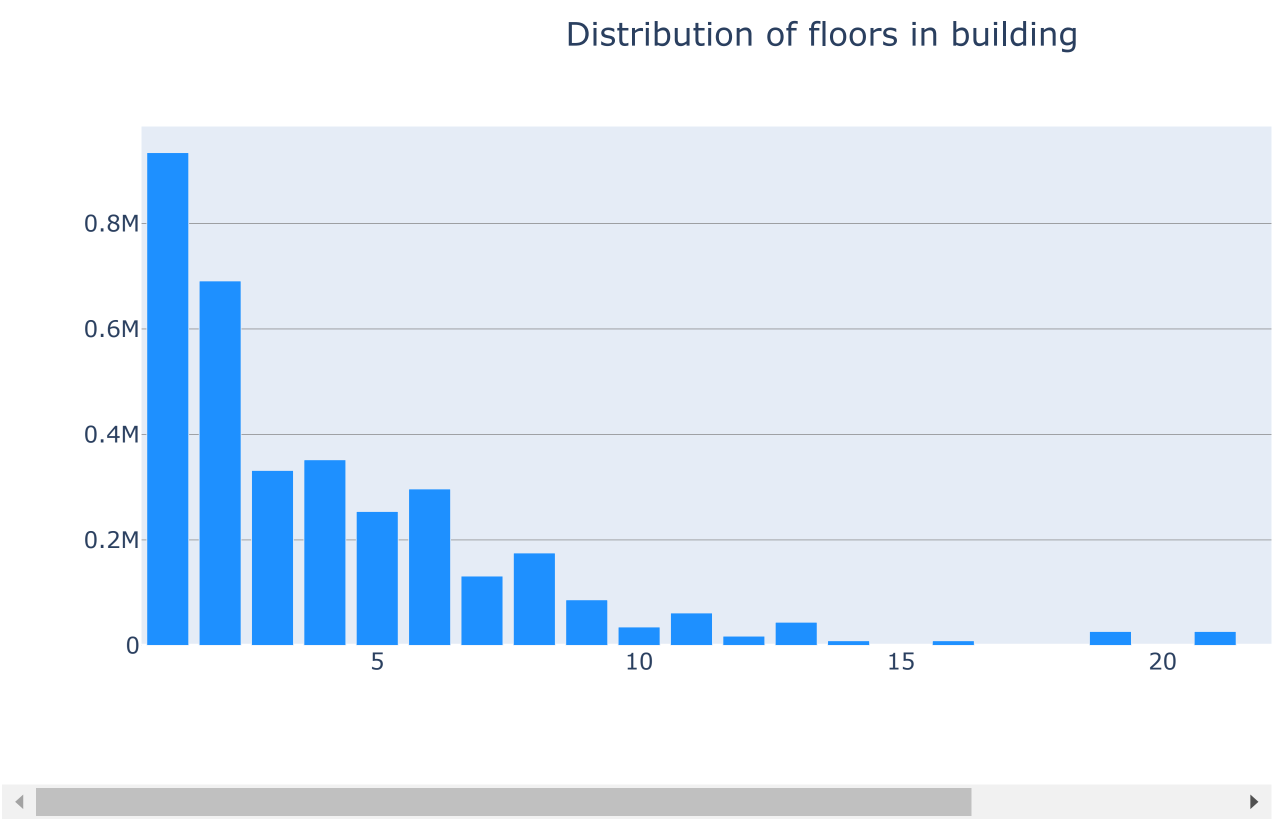
In [17]:
```
data_index = building_data["primary_use"].value_counts()
#data_index = data_index.sort_index()
trace = go.Bar(
    x=data_index.index,
    y=data_index.values,
    marker=dict(
        color="#1E90FF",
    ),
)

layout = go.Layout(
    title=go.layout.Title(
        text="Distribution of primary use of Buildings",
        x=0.5
    ),
    font=dict(size=14),
    width=1000,
    height=500,
)

data = [trace]
fig = go.Figure(data=data, layout=layout)
py.iplot(fig, filename="meter")
```



Education is the one with most number of primary usage followed by office adn entertainment

Floor count in the building

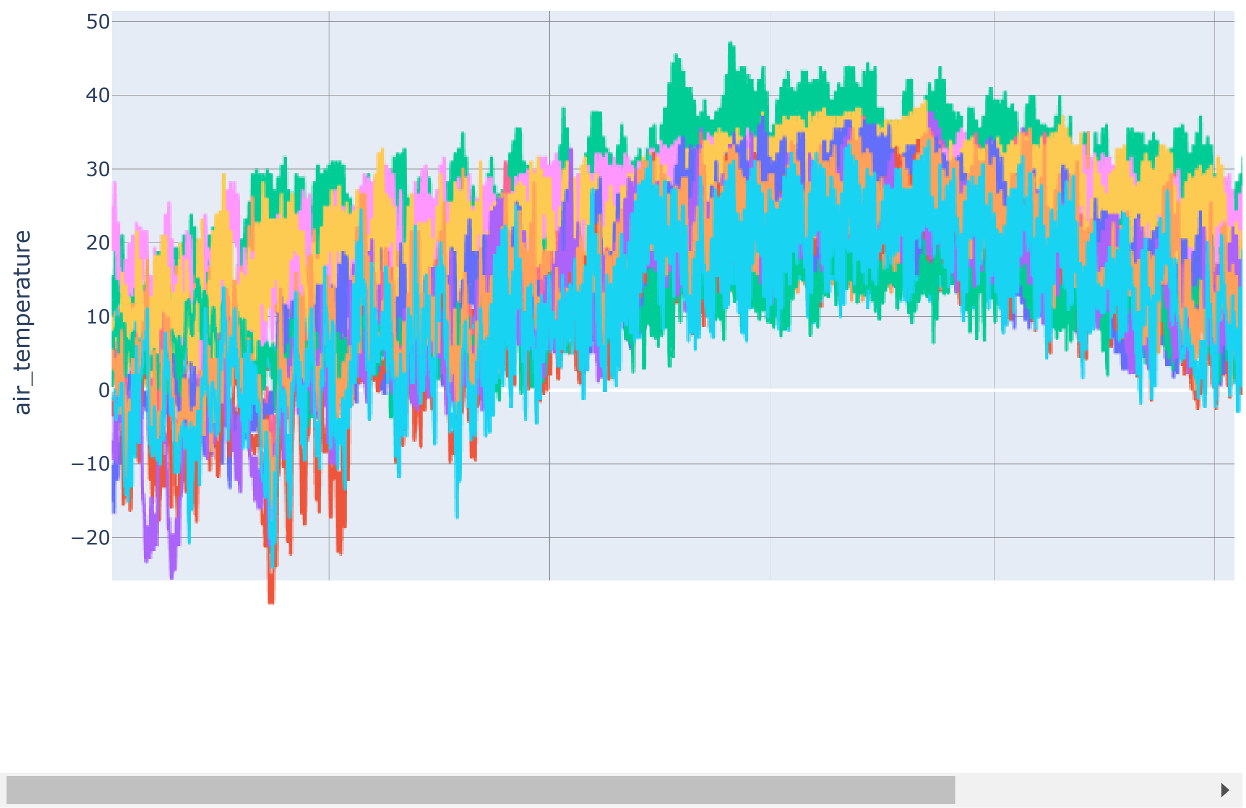
In [18]:
```
dataset = train["floor_count"].value_counts()

trace = go.Bar(
    x=dataset.index,
    y=dataset.values,
    marker=dict(
        color="#1E90FF",
    ),
)

layout = go.Layout(
    title=go.layout.Title(
        text="Distribution of floors in building",
        x=0.5
    ),
    font=dict(size=14),
    width=1000,
    height=500,
)

data = [trace]
fig = go.Figure(data=data, layout=layout)
py.iplot(fig, filename="meter")
```



Distribution plots for Weather

Air temperature distribution

In [19]:
```
fig = px.line(weather_train, x='timestamp', y='air_temperature', color='site_id')
fig.show()
```



Looking at the graph it seems that the temperature increases in all the sites towards the middle of the year and decreases at the end of year.

In [6]:
```
train[:]
```

Out[6]:

	building_id	meter	timestamp	meter_reading	site_id	primary_use	square_feet	year_built	floor_count
0	0	0	2016-01-01 00:00:00	0.000000	0	Education	7432	2008.0	NaN
1	1	0	2016-01-01 00:00:00	0.000000	0	Education	2720	2004.0	NaN
2	2	0	2016-01-01 00:00:00	0.000000	0	Education	5376	1991.0	NaN
3	3	0	2016-01-01 00:00:00	0.000000	0	Education	23685	2002.0	NaN
4	4	0	2016-01-01 00:00:00	0.000000	0	Education	116607	1975.0	NaN
5	5	0	2016-01-01 00:00:00	0.000000	0	Education	8000	2000.0	NaN
6	6	0	2016-01-01 00:00:00	0.000000	0	Lodging/residential	27926	1981.0	NaN
7	7	0	2016-01-01 00:00:00	0.000000	0	Education	121074	1989.0	NaN
8	8	0	2016-01-01 00:00:00	0.000000	0	Education	60809	2003.0	NaN
9	9	0	2016-01-01 00:00:00	0.000000	0	Office	27000	2010.0	NaN
10	10	0	2016-01-01 00:00:00	0.000000	0	Entertainment/public assembly	370773	1991.0	NaN
11	11	0	2016-01-01 00:00:00	0.000000	0	Education	49073	1968.0	NaN
12	12	0	2016-01-01 00:00:00	0.000000	0	Lodging/residential	37100	1999.0	NaN
13	13	0	2016-01-01 00:00:00	0.000000	0	Education	99380	2000.0	NaN
14	14	0	2016-01-01 00:00:00	0.000000	0	Education	86250	2013.0	NaN
15	15	0	2016-01-01 00:00:00	0.000000	0	Office	83957	1974.0	NaN
16	16	0	2016-01-01 00:00:00	0.000000	0	Education	54644	1996.0	NaN
17	17	0	2016-01-01 00:00:00	0.000000	0	Office	15250	1980.0	NaN
18	18	0	2016-01-01 00:00:00	0.000000	0	Education	111891	1996.0	NaN
19	19	0	2016-01-01 00:00:00	0.000000	0	Office	18717	2004.0	NaN
20	20	0	2016-01-01 00:00:00	0.000000	0	Education	110272	1977.0	NaN

	building_id	meter	timestamp	meter_reading	site_id	primary_use	square_feet	year_built	floor_count
21	21	0	2016-01-01 00:00:00	0.000000	0	Office	7043	1990.0	NaN
22	22	0	2016-01-01 00:00:00	0.000000	0	Education	3569	1996.0	NaN
23	23	0	2016-01-01 00:00:00	0.000000	0	Education	130885	1985.0	NaN
24	24	0	2016-01-01 00:00:00	0.000000	0	Education	105545	2001.0	NaN
25	25	0	2016-01-01 00:00:00	0.000000	0	Office	103286	1969.0	NaN
26	26	0	2016-01-01 00:00:00	0.000000	0	Office	26953	2005.0	NaN
27	27	0	2016-01-01 00:00:00	0.000000	0	Lodging/residential	59200	1999.0	NaN
28	28	0	2016-01-01 00:00:00	0.000000	0	Office	52957	2016.0	NaN
29	30	0	2016-01-01 00:00:00	0.000000	0	Education	93897	1999.0	NaN
...	...	...	...	...	...	...	...	...	...
20216070	1427	0	2016-12-31 23:00:00	145.199997	15	Education	180625	1933.0	NaN
20216071	1427	2	2016-12-31 23:00:00	3006.820068	15	Education	180625	1933.0	NaN
20216072	1428	0	2016-12-31 23:00:00	38.724998	15	Education	28432	1933.0	NaN
20216073	1429	0	2016-12-31 23:00:00	27.775000	15	Education	40461	2002.0	NaN
20216074	1430	2	2016-12-31 23:00:00	318.567993	15	Office	53303	1981.0	NaN
20216075	1431	0	2016-12-31 23:00:00	87.949997	15	Public services	111360	2000.0	NaN
20216076	1431	2	2016-12-31 23:00:00	426.339996	15	Public services	111360	2000.0	NaN
20216077	1432	0	2016-12-31 23:00:00	403.450012	15	Education	160673	1968.0	NaN
20216078	1433	0	2016-12-31 23:00:00	41.349998	15	Education	28084	1913.0	NaN
20216079	1433	2	2016-12-31 23:00:00	3173.879883	15	Education	28084	1913.0	NaN
20216080	1434	0	2016-12-31 23:00:00	70.724998	15	Education	33148	1967.0	NaN
20216081	1434	2	2016-12-31 23:00:00	259.072998	15	Education	33148	1967.0	NaN

	building_id	meter	timestamp	meter_reading	site_id	primary_use	square_feet	year_built	floor_count
20216082	1435	0	2016-12-31 23:00:00	4.725000	15	Education	9552	1961.0	NaN
20216083	1436	0	2016-12-31 23:00:00	11.600000	15	Manufacturing/industrial	11302	1937.0	NaN
20216084	1436	2	2016-12-31 23:00:00	1274.660034	15	Manufacturing/industrial	11302	1937.0	NaN
20216085	1437	0	2016-12-31 23:00:00	195.925003	15	Education	111518	1968.0	NaN
20216086	1437	2	2016-12-31 23:00:00	1518.920044	15	Education	111518	1968.0	NaN
20216087	1438	0	2016-12-31 23:00:00	100.675003	15	Education	108971	1990.0	NaN
20216088	1438	2	2016-12-31 23:00:00	852.770020	15	Education	108971	1990.0	NaN
20216089	1439	0	2016-12-31 23:00:00	167.399994	15	Education	56497	1957.0	NaN
20216090	1440	0	2016-12-31 23:00:00	154.750000	15	Lodging/residential	150294	1987.0	NaN
20216091	1441	0	2016-12-31 23:00:00	242.925003	15	Education	30143	1951.0	NaN
20216092	1442	0	2016-12-31 23:00:00	59.400002	15	Public services	99541	1993.0	NaN
20216093	1442	2	2016-12-31 23:00:00	55.624100	15	Public services	99541	1993.0	NaN
20216094	1443	0	2016-12-31 23:00:00	64.949997	15	Education	40311	1913.0	NaN
20216095	1444	0	2016-12-31 23:00:00	8.750000	15	Entertainment/public assembly	19619	1914.0	NaN
20216096	1445	0	2016-12-31 23:00:00	4.825000	15	Education	4298	NaN	NaN
20216097	1446	0	2016-12-31 23:00:00	0.000000	15	Entertainment/public assembly	11265	1997.0	NaN
20216098	1447	0	2016-12-31 23:00:00	159.574997	15	Lodging/residential	29775	2001.0	NaN
20216099	1448	0	2016-12-31 23:00:00	2.850000	15	Office	92271	2001.0	NaN

20216100 rows × 16 columns

In [7]:
```
train1 = train
```

In [8]:
```
train1[:]
```

Out[8]:

	building_id	meter	timestamp	meter_reading	site_id	primary_use	square_feet	year_built	floor_count
0	0	0	2016-01-01 00:00:00	0.000000	0	Education	7432	2008.0	NaN
1	1	0	2016-01-01 00:00:00	0.000000	0	Education	2720	2004.0	NaN
2	2	0	2016-01-01 00:00:00	0.000000	0	Education	5376	1991.0	NaN
3	3	0	2016-01-01 00:00:00	0.000000	0	Education	23685	2002.0	NaN
4	4	0	2016-01-01 00:00:00	0.000000	0	Education	116607	1975.0	NaN
5	5	0	2016-01-01 00:00:00	0.000000	0	Education	8000	2000.0	NaN
6	6	0	2016-01-01 00:00:00	0.000000	0	Lodging/residential	27926	1981.0	NaN
7	7	0	2016-01-01 00:00:00	0.000000	0	Education	121074	1989.0	NaN
8	8	0	2016-01-01 00:00:00	0.000000	0	Education	60809	2003.0	NaN
9	9	0	2016-01-01 00:00:00	0.000000	0	Office	27000	2010.0	NaN
10	10	0	2016-01-01 00:00:00	0.000000	0	Entertainment/public assembly	370773	1991.0	NaN
11	11	0	2016-01-01 00:00:00	0.000000	0	Education	49073	1968.0	NaN
12	12	0	2016-01-01 00:00:00	0.000000	0	Lodging/residential	37100	1999.0	NaN
13	13	0	2016-01-01 00:00:00	0.000000	0	Education	99380	2000.0	NaN
14	14	0	2016-01-01 00:00:00	0.000000	0	Education	86250	2013.0	NaN
15	15	0	2016-01-01 00:00:00	0.000000	0	Office	83957	1974.0	NaN
16	16	0	2016-01-01 00:00:00	0.000000	0	Education	54644	1996.0	NaN
17	17	0	2016-01-01 00:00:00	0.000000	0	Office	15250	1980.0	NaN
18	18	0	2016-01-01 00:00:00	0.000000	0	Education	111891	1996.0	NaN
19	19	0	2016-01-01 00:00:00	0.000000	0	Office	18717	2004.0	NaN
20	20	0	2016-01-01 00:00:00	0.000000	0	Education	110272	1977.0	NaN

	building_id	meter	timestamp	meter_reading	site_id	primary_use	square_feet	year_built	floor_count
21	21	0	2016-01-01 00:00:00	0.000000	0	Office	7043	1990.0	NaN
22	22	0	2016-01-01 00:00:00	0.000000	0	Education	3569	1996.0	NaN
23	23	0	2016-01-01 00:00:00	0.000000	0	Education	130885	1985.0	NaN
24	24	0	2016-01-01 00:00:00	0.000000	0	Education	105545	2001.0	NaN
25	25	0	2016-01-01 00:00:00	0.000000	0	Office	103286	1969.0	NaN
26	26	0	2016-01-01 00:00:00	0.000000	0	Office	26953	2005.0	NaN
27	27	0	2016-01-01 00:00:00	0.000000	0	Lodging/residential	59200	1999.0	NaN
28	28	0	2016-01-01 00:00:00	0.000000	0	Office	52957	2016.0	NaN
29	30	0	2016-01-01 00:00:00	0.000000	0	Education	93897	1999.0	NaN
...	...	...	...	...	...	...	...	...	...
20216070	1427	0	2016-12-31 23:00:00	145.199997	15	Education	180625	1933.0	NaN
20216071	1427	2	2016-12-31 23:00:00	3006.820068	15	Education	180625	1933.0	NaN
20216072	1428	0	2016-12-31 23:00:00	38.724998	15	Education	28432	1933.0	NaN
20216073	1429	0	2016-12-31 23:00:00	27.775000	15	Education	40461	2002.0	NaN
20216074	1430	2	2016-12-31 23:00:00	318.567993	15	Office	53303	1981.0	NaN
20216075	1431	0	2016-12-31 23:00:00	87.949997	15	Public services	111360	2000.0	NaN
20216076	1431	2	2016-12-31 23:00:00	426.339996	15	Public services	111360	2000.0	NaN
20216077	1432	0	2016-12-31 23:00:00	403.450012	15	Education	160673	1968.0	NaN
20216078	1433	0	2016-12-31 23:00:00	41.349998	15	Education	28084	1913.0	NaN
20216079	1433	2	2016-12-31 23:00:00	3173.879883	15	Education	28084	1913.0	NaN
20216080	1434	0	2016-12-31 23:00:00	70.724998	15	Education	33148	1967.0	NaN
20216081	1434	2	2016-12-31 23:00:00	259.072998	15	Education	33148	1967.0	NaN

	building_id	meter	timestamp	meter_reading	site_id	primary_use	square_feet	year_built	floor_count
20216082	1435	0	2016-12-31 23:00:00	4.725000	15	Education	9552	1961.0	NaN
20216083	1436	0	2016-12-31 23:00:00	11.600000	15	Manufacturing/industrial	11302	1937.0	NaN
20216084	1436	2	2016-12-31 23:00:00	1274.660034	15	Manufacturing/industrial	11302	1937.0	NaN
20216085	1437	0	2016-12-31 23:00:00	195.925003	15	Education	111518	1968.0	NaN
20216086	1437	2	2016-12-31 23:00:00	1518.920044	15	Education	111518	1968.0	NaN
20216087	1438	0	2016-12-31 23:00:00	100.675003	15	Education	108971	1990.0	NaN
20216088	1438	2	2016-12-31 23:00:00	852.770020	15	Education	108971	1990.0	NaN
20216089	1439	0	2016-12-31 23:00:00	167.399994	15	Education	56497	1957.0	NaN
20216090	1440	0	2016-12-31 23:00:00	154.750000	15	Lodging/residential	150294	1987.0	NaN
20216091	1441	0	2016-12-31 23:00:00	242.925003	15	Education	30143	1951.0	NaN
20216092	1442	0	2016-12-31 23:00:00	59.400002	15	Public services	99541	1993.0	NaN
20216093	1442	2	2016-12-31 23:00:00	55.624100	15	Public services	99541	1993.0	NaN
20216094	1443	0	2016-12-31 23:00:00	64.949997	15	Education	40311	1913.0	NaN
20216095	1444	0	2016-12-31 23:00:00	8.750000	15	Entertainment/public assembly	19619	1914.0	NaN
20216096	1445	0	2016-12-31 23:00:00	4.825000	15	Education	4298	NaN	NaN
20216097	1446	0	2016-12-31 23:00:00	0.000000	15	Entertainment/public assembly	11265	1997.0	NaN
20216098	1447	0	2016-12-31 23:00:00	159.574997	15	Lodging/residential	29775	2001.0	NaN
20216099	1448	0	2016-12-31 23:00:00	2.850000	15	Office	92271	2001.0	NaN

20216100 rows × 16 columns

In [6]:
```
train = train[["building_id",'meter','timestamp','site_id','primary_use','square_feet','year_built',
'floor_count','air_temperature','cloud_coverage','dew_temperature','precip_depth_1_hr','sea_level_pr
essure','wind_direction','wind_speed','meter_reading']]
```

In [8]:
```
train[:2]
```

Out[8]:

	building_id	meter	timestamp	site_id	primary_use	square_feet	year_built	floor_count	air_temperature	cloud_coverage
0	0	0	2016-01-01 00:00:00	0	Education	7432	2008.0	NaN	25.0	6.0
1	1	0	2016-01-01 00:00:00	0	Education	2720	2004.0	NaN	25.0	6.0

In [25]:
```
test[:2]
```

Out[25]:

	row_id	building_id	meter	timestamp	site_id	primary_use	square_feet	year_built	floor_count	air_temperature	cloud_c
0	0	0	0	2017-01-01 00:00:00	0	Education	7432	2008.0	NaN	17.796875	
1	1	1	0	2017-01-01 00:00:00	0	Education	2720	2004.0	NaN	17.796875	

In [11]:
```
train2 = train1
```

In [12]:
```
train2[:]
```

Out[12]:

	building_id	meter	timestamp	site_id	primary_use	square_feet	year_built	floor_count	air_temperature
0	0	0	2016-01-01 00:00:00	0	Education	7432	2008.0	NaN	25.000000
1	1	0	2016-01-01 00:00:00	0	Education	2720	2004.0	NaN	25.000000
2	2	0	2016-01-01 00:00:00	0	Education	5376	1991.0	NaN	25.000000
3	3	0	2016-01-01 00:00:00	0	Education	23685	2002.0	NaN	25.000000
4	4	0	2016-01-01 00:00:00	0	Education	116607	1975.0	NaN	25.000000
5	5	0	2016-01-01 00:00:00	0	Education	8000	2000.0	NaN	25.000000
6	6	0	2016-01-01 00:00:00	0	Lodging/residential	27926	1981.0	NaN	25.000000
7	7	0	2016-01-01 00:00:00	0	Education	121074	1989.0	NaN	25.000000
8	8	0	2016-01-01 00:00:00	0	Education	60809	2003.0	NaN	25.000000
9	9	0	2016-01-01 00:00:00	0	Office	27000	2010.0	NaN	25.000000
10	10	0	2016-01-01 00:00:00	0	Entertainment/public assembly	370773	1991.0	NaN	25.000000
11	11	0	2016-01-01 00:00:00	0	Education	49073	1968.0	NaN	25.000000
12	12	0	2016-01-01 00:00:00	0	Lodging/residential	37100	1999.0	NaN	25.000000
13	13	0	2016-01-01 00:00:00	0	Education	99380	2000.0	NaN	25.000000
14	14	0	2016-01-01 00:00:00	0	Education	86250	2013.0	NaN	25.000000
15	15	0	2016-01-01 00:00:00	0	Office	83957	1974.0	NaN	25.000000
16	16	0	2016-01-01 00:00:00	0	Education	54644	1996.0	NaN	25.000000
17	17	0	2016-01-01 00:00:00	0	Office	15250	1980.0	NaN	25.000000
18	18	0	2016-01-01 00:00:00	0	Education	111891	1996.0	NaN	25.000000
19	19	0	2016-01-01 00:00:00	0	Office	18717	2004.0	NaN	25.000000
20	20	0	2016-01-01 00:00:00	0	Education	110272	1977.0	NaN	25.000000

	building_id	meter	timestamp	site_id	primary_use	square_feet	year_built	floor_count	air_temperature
21	21	0	2016-01-01 00:00:00	0	Office	7043	1990.0	NaN	25.000000
22	22	0	2016-01-01 00:00:00	0	Education	3569	1996.0	NaN	25.000000
23	23	0	2016-01-01 00:00:00	0	Education	130885	1985.0	NaN	25.000000
24	24	0	2016-01-01 00:00:00	0	Education	105545	2001.0	NaN	25.000000
25	25	0	2016-01-01 00:00:00	0	Office	103286	1969.0	NaN	25.000000
26	26	0	2016-01-01 00:00:00	0	Office	26953	2005.0	NaN	25.000000
27	27	0	2016-01-01 00:00:00	0	Lodging/residential	59200	1999.0	NaN	25.000000
28	28	0	2016-01-01 00:00:00	0	Office	52957	2016.0	NaN	25.000000
29	30	0	2016-01-01 00:00:00	0	Education	93897	1999.0	NaN	25.000000
...	...	...	...	...	...	...	...	...	...
20216070	1427	0	2016-12-31 23:00:00	15	Education	180625	1933.0	NaN	1.700195
20216071	1427	2	2016-12-31 23:00:00	15	Education	180625	1933.0	NaN	1.700195
20216072	1428	0	2016-12-31 23:00:00	15	Education	28432	1933.0	NaN	1.700195
20216073	1429	0	2016-12-31 23:00:00	15	Education	40461	2002.0	NaN	1.700195
20216074	1430	2	2016-12-31 23:00:00	15	Office	53303	1981.0	NaN	1.700195
20216075	1431	0	2016-12-31 23:00:00	15	Public services	111360	2000.0	NaN	1.700195
20216076	1431	2	2016-12-31 23:00:00	15	Public services	111360	2000.0	NaN	1.700195
20216077	1432	0	2016-12-31 23:00:00	15	Education	160673	1968.0	NaN	1.700195
20216078	1433	0	2016-12-31 23:00:00	15	Education	28084	1913.0	NaN	1.700195
20216079	1433	2	2016-12-31 23:00:00	15	Education	28084	1913.0	NaN	1.700195
20216080	1434	0	2016-12-31 23:00:00	15	Education	33148	1967.0	NaN	1.700195
20216081	1434	2	2016-12-31 23:00:00	15	Education	33148	1967.0	NaN	1.700195

	building_id	meter	timestamp	site_id	primary_use	square_feet	year_built	floor_count	air_temperature
20216082	1435	0	2016-12-31 23:00:00	15	Education	9552	1961.0	NaN	1.700195
20216083	1436	0	2016-12-31 23:00:00	15	Manufacturing/industrial	11302	1937.0	NaN	1.700195
20216084	1436	2	2016-12-31 23:00:00	15	Manufacturing/industrial	11302	1937.0	NaN	1.700195
20216085	1437	0	2016-12-31 23:00:00	15	Education	111518	1968.0	NaN	1.700195
20216086	1437	2	2016-12-31 23:00:00	15	Education	111518	1968.0	NaN	1.700195
20216087	1438	0	2016-12-31 23:00:00	15	Education	108971	1990.0	NaN	1.700195
20216088	1438	2	2016-12-31 23:00:00	15	Education	108971	1990.0	NaN	1.700195
20216089	1439	0	2016-12-31 23:00:00	15	Education	56497	1957.0	NaN	1.700195
20216090	1440	0	2016-12-31 23:00:00	15	Lodging/residential	150294	1987.0	NaN	1.700195
20216091	1441	0	2016-12-31 23:00:00	15	Education	30143	1951.0	NaN	1.700195
20216092	1442	0	2016-12-31 23:00:00	15	Public services	99541	1993.0	NaN	1.700195
20216093	1442	2	2016-12-31 23:00:00	15	Public services	99541	1993.0	NaN	1.700195
20216094	1443	0	2016-12-31 23:00:00	15	Education	40311	1913.0	NaN	1.700195
20216095	1444	0	2016-12-31 23:00:00	15	Entertainment/public assembly	19619	1914.0	NaN	1.700195
20216096	1445	0	2016-12-31 23:00:00	15	Education	4298	NaN	NaN	1.700195
20216097	1446	0	2016-12-31 23:00:00	15	Entertainment/public assembly	11265	1997.0	NaN	1.700195
20216098	1447	0	2016-12-31 23:00:00	15	Lodging/residential	29775	2001.0	NaN	1.700195
20216099	1448	0	2016-12-31 23:00:00	15	Office	92271	2001.0	NaN	1.700195

20216100 rows × 16 columns

In [9]:
```
train.drop(['timestamp','primary_use','floor_count','precip_depth_1_hr'], axis=1, inplace=True)
```

In [10]:
```
train[:2]
```

Out[10]:

	building_id	meter	site_id	square_feet	year_built	air_temperature	cloud_coverage	dew_temperature	sea_level_pressure
0	0	0	0	7432	2008.0	25.0	6.0	20.0	1019.5
1	1	0	0	2720	2004.0	25.0	6.0	20.0	1019.5

In [20]:
```
train.dropna(inplace=True)
```

In [21]:
```
train.isnull().any()
```

Out[21]:
```
building_id           False
meter                 False
site_id               False
square_feet           False
year_built            False
air_temperature       False
cloud_coverage        False
dew_temperature       False
sea_level_pressure    False
wind_direction        False
wind_speed            False
meter_reading         False
dtype: bool
```

In [22]:
```
train.shape
```

Out[22]:
```
(3742809, 12)
```

In [24]:
```
import pandas
from keras.models import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasRegressor
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
```

In [25]:
```
# load dataset
dataframe = train
dataset = dataframe.values
# split into input (X) and output (Y) variables
X = dataset[:,0:11]
Y = dataset[:,11]
```

In [26]:
```
X[120,0:11]
```

Out[26]:
```
array([ 1.69000000e+02,  0.00000000e+00,  2.00000000e+00,  1.79559000e+05,
        2.00600000e+03,  1.56015625e+01,  6.00000000e+00, -5.60156250e+00,
        1.01550000e+03,  2.70000000e+02,  3.59960938e+00])
```

In [27]:
```
Y[120]
```

Out[27]:
```
468.7099914550781
```

In [29]:
```
# define base model
def baseline_model():
        # create model
        model = Sequential()
        model.add(Dense(11, input_dim=11, kernel_initializer='normal', activation='relu'))
        model.add(Dense(6, kernel_initializer='normal', activation='relu'))
        model.add(Dense(1, kernel_initializer='normal'))
        # Compile model
        model.compile(loss='mean_squared_error', optimizer='adam')
        return model
# evaluate model
estimator = KerasRegressor(build_fn=baseline_model, epochs=10, verbose=1)
kfold = KFold(n_splits=2)
results = cross_val_score(estimator, X, Y, cv=kfold)
print("Baseline: %.2f (%.2f) MSE" % (results.mean(), results.std()))
```

```
Epoch 1/10
1871404/1871404 [==============================] - 42s 22us/step - loss: 382694.6372
Epoch 2/10
1871404/1871404 [==============================] - 41s 22us/step - loss: 363724.9191
Epoch 3/10
1871404/1871404 [==============================] - 41s 22us/step - loss: 359972.0451
Epoch 4/10
1871404/1871404 [==============================] - 42s 22us/step - loss: 356121.7644
Epoch 5/10
1871404/1871404 [==============================] - 42s 22us/step - loss: 348780.4941
Epoch 6/10
1871404/1871404 [==============================] - 42s 22us/step - loss: 342920.5658
Epoch 7/10
1871404/1871404 [==============================] - 42s 22us/step - loss: 344260.4191
Epoch 8/10
1871404/1871404 [==============================] - 42s 22us/step - loss: 343940.7274
Epoch 9/10
1871404/1871404 [==============================] - 42s 22us/step - loss: 329363.4497
Epoch 10/10
1871404/1871404 [==============================] - 42s 22us/step - loss: 327680.0215
1871405/1871405 [==============================] - 22s 12us/step
Epoch 1/10
1871405/1871405 [==============================] - 44s 23us/step - loss: 303178.8009
Epoch 2/10
1871405/1871405 [==============================] - 44s 23us/step - loss: 284768.2193
Epoch 3/10
1871405/1871405 [==============================] - 43s 23us/step - loss: 274759.9143
Epoch 4/10
1871405/1871405 [==============================] - 43s 23us/step - loss: 270544.8679
Epoch 5/10
1871405/1871405 [==============================] - 43s 23us/step - loss: 268217.8223
Epoch 6/10
1871405/1871405 [==============================] - 44s 23us/step - loss: 266581.6940
Epoch 7/10
1871405/1871405 [==============================] - 43s 23us/step - loss: 278975.8212
Epoch 8/10
1871405/1871405 [==============================] - 44s 23us/step - loss: 283684.6905
Epoch 9/10
1871405/1871405 [==============================] - 43s 23us/step - loss: 268706.6465
Epoch 10/10
1871405/1871405 [==============================] - 43s 23us/step - loss: 266053.2907
1871404/1871404 [==============================] - 22s 12us/step
Baseline: -321538.28 (45654.86) MSE
```

In [57]:
```
model.summary()
```

```
Model: "sequential_11"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_31 (Dense)             (None, 11)                132       
_________________________________________________________________
dense_32 (Dense)             (None, 6)                 72        
_________________________________________________________________
dense_33 (Dense)             (None, 1)                 7         
=================================================================
Total params: 211
Trainable params: 211
Non-trainable params: 0
_________________________________________________________________
```

In [62]:
```
print("Baseline: %.2f (%.2f) MSE (%.2f) RMSE" % (results.mean(), results.std(), (results.std()**0.5
)))
```

```
Baseline: -321538.28 (45654.86) MSE (213.67) RMSE
```

In [56]:
```
model = baseline_model()
```

In [59]:
```
example_batch = test[:100]
example_result = model.predict(example_batch)
example_result
```

Out[59]:
```
array([[-0.67818165],
       [-0.49512476],
       [-0.6679115 ],
       [-0.70683396],
       [-0.8525908 ],
       [-0.67858166],
       [-0.7106684 ],
       [-0.8605795 ],
       [-0.86102057],
       [-0.7652935 ],
       [-0.711642  ],
       [-0.7120826 ],
       [-1.260273  ],
       [-0.742005  ],
       [-0.72595024],
       [-0.82563806],
       [-0.82607794],
       [-0.8057761 ],
       [-0.80621624],
       [-0.79764485],
       [-0.7980845 ],
       [-0.7528267 ],
       [-0.687731  ],
       [-0.84411   ],
       [-0.69543177],
       [-0.8389976 ],
       [-0.6703227 ],
       [-0.5854686 ],
       [-0.87221384],
       [-0.8331115 ],
       [-0.82579255],
       [-0.70713913],
       [-0.75793886],
       [-0.749547  ],
       [-0.7499877 ],
       [-0.84584475],
       [-0.84628344],
       [-0.8128576 ],
       [-0.81329656],
       [-0.76157403],
       [-0.7620139 ],
       [-0.7401247 ],
       [-0.72401667],
       [-0.7302706 ],
       [-0.7282561 ],
       [-0.7464775 ],
       [-0.7462499 ],
       [-0.6825325 ],
       [-0.7606276 ],
       [-0.70349103],
       [-0.8069222 ],
       [-1.019269  ],
       [-0.7520149 ],
       [-0.7524543 ],
       [-1.085437  ],
       [-0.6853201 ],
       [-0.6750611 ],
       [-0.7102166 ],
       [-0.67251635],
       [-0.6938688 ],
       [-0.6538059 ],
       [-0.65421367],
       [-1.2804317 ],
       [-0.8249192 ],
       [-0.7946999 ],
       [-0.66846347],
       [-0.68477106],
       [-0.68521124],
       [-0.754251  ],
       [-0.7624688 ],
```

```
       [-0.7911191 ],
       [-0.78674984],
       [-0.7860377 ],
       [-0.78647757],
       [-0.7175075 ],
       [-0.72376156],
       [-0.72174764],
       [-0.7399703 ],
       [-0.7397412 ],
       [-0.71188104],
       [-0.6897546 ],
       [-1.4357147 ],
       [-1.2747917 ],
       [-1.2753181 ],
       [-1.2741175 ],
       [-1.2743263 ],
       [-1.2747421 ],
       [-1.273324  ],
       [-0.78300095],
       [-0.7834425 ],
       [-0.8594985 ],
       [-0.8599396 ],
       [-0.68746865],
       [-0.7744837 ],
       [-0.7749238 ],
       [-0.7104491 ],
       [-0.90134716],
       [-0.53398895],
       [-0.66499996],
       [-0.6654401 ]], dtype=float32)
```

In [32]:
```
test[0:2]
```

Out[32]:

	row_id	building_id	meter	timestamp	site_id	primary_use	square_feet	year_built	floor_count	air_temperature	cloud_c
0	0	0	0	2017-01-01 00:00:00	0	Education	7432	2008.0	NaN	17.796875	
1	1	1	0	2017-01-01 00:00:00	0	Education	2720	2004.0	NaN	17.796875	

In [33]:
```
test.drop(['row_id','timestamp','primary_use','floor_count','precip_depth_1_hr'], axis=1, inplace=True)
```

In [34]:
```
test[0:2]
```

Out[34]:

	building_id	meter	site_id	square_feet	year_built	air_temperature	cloud_coverage	dew_temperature	sea_level_pressure
0	0	0	0	7432	2008.0	17.796875	4.0	11.703125	1021.5
1	1	0	0	2720	2004.0	17.796875	4.0	11.703125	1021.5

In [35]:
```
test.isnull().any()
```

Out[35]:
```
building_id           False
meter                 False
site_id               False
square_feet           False
year_built             True
air_temperature        True
cloud_coverage         True
dew_temperature        True
sea_level_pressure     True
wind_direction         True
wind_speed             True
dtype: bool
```

In [36]:
```
test.dropna(inplace=True)
```

In [37]:
```
test.isnull().any()
```

Out[37]:
```
building_id           False
meter                 False
site_id               False
square_feet           False
year_built            False
air_temperature       False
cloud_coverage        False
dew_temperature       False
sea_level_pressure    False
wind_direction        False
wind_speed            False
dtype: bool
```

In [38]:
```
test.shape
```

Out[38]:
```
(7486737, 11)
```

In [42]:
```
test[0:2]
```

Out[42]:

	building_id	meter	site_id	square_feet	year_built	air_temperature	cloud_coverage	dew_temperature	sea_level_pressure
0	0	0	0	7432	2008.0	17.796875	4.0	11.703125	1021.5
1	1	0	0	2720	2004.0	17.796875	4.0	11.703125	1021.5

In [51]:
```
from sklearn.metrics import accuracy_score
prediction = estimator.predict(Xnew)
print(prediction)
```

```
1/1 [==============================] - 0s 684us/step
[-5711.1846]
```

In [ ]: