Heatmap_np

January 12, 2021

1 Heatmap determining data quality for Energy Production

[7]:
```
#Import modules:
import numpy as np
import pandas as pd
from tqdm import tqdm
import matplotlib.pyplot as plt
import glob
import seaborn as sns

from sklearn.svm import SVR
from sklearn.model_selection import train_test_split
from statsmodels.tsa.arima_model import ARMA
```

[8]:
```
#functions
def nParse3(n):
    """output a string that is 3 decimals long. Levy en Jefry kunnen uitleg␣
 ↪geven"""
    number = str(n)
    if len(number) == 1:
        number = "00" + number
    elif len(number) == 2:
        number = "0" + number
    elif len(number) == 3:
        number = number
    return str(number)

def check_eq_w_range(Numpy_Sum,Excel_Sum,abs_diff):
    """Check if a value is equal within a certain range."""
    if abs(Numpy_Sum-Excel_Sum) < abs_diff:
        return True
    else:
        return False

def open_file_append_sentence(msg):
    """open the file, append the sentence, and then close the file."""
    with open("DataCheck_info_NEW.txt",'a') as f:
```

```
        f.write(str(msg))
        f.close()
```

Heatmap #1 (mean, max, min)

[ ]:
```
#Select the path to the datasets and the sheets to be collected.
path = '/home/18062814/notebooks/zero/DATA/'

sheets =␣
 ↪['energyImmersion','energyHeatpump','smartMeter','energyWtwReg','solar']

#Define a function with which to compute the difference in timestep per sheet.
def add_diff(df):
    try:
        df[len(df.columns)+1] = abs(df[0].diff()-300).fillna(0)
    except:
        df[len(df.columns)+1] = 0
    return df

#Create an empty dictionary object; Key = sheetname, Value = house number.
#Start a for loop to add dictionaries to it for each house; Key = house number,␣
 ↪Value = dataframe containing data of selected sheet.
D = {}

for i in range(1,121):
    for sheet in sheets:
        df = pd.DataFrame(np.load(path + sheet + '_' + nParse3(i) + '.npy'))
        df = add_diff(df)

        col_names = list(df.columns)
        col_names[-1] = "diff_ts"
        df.columns = col_names

        if sheet in D.keys():
            D[sheet][nParse3(i)] = df
        else:
            D[sheet] = {nParse3(i):df}

#Create an empty dataframe object; containing the mean, max and min value of␣
 ↪diff_ts per sheet for each house.
#Loop over the data stored in dictionary.
#data_df_mean = pd.DataFrame()
#data_df_min = pd.DataFrame()
#data_df_max = pd.DataFrame()

data_list = []
```

```
for i in range(1, 121):
    for sheet in sheets:
        data = D[sheet][nParse3(i)]['diff_ts']

        row = [sheet,i,data.mean(),data.min(),data.max()]
        data_list.append(row)


        #data_df_mean[sheet + '_ts_mean'] = data.mean()
        #data_df_min[sheet + '_ts_min'] = nata.min()
        #data_df_max[sheet + '_ts_max'] = data.max()

data_df = pd.
 ↪DataFrame(data_list,columns=["Sheet","HouseNumber","MEAN","MIN","MAX"])
data_df_mean = data_df.pivot('HouseNumber','Sheet','MEAN')
data_df_min = data_df.pi




# %matplotlib notebook
# plt.subplots(figsize=(30,7))
# st_df = st_df.pivot(0,1,3)
# ax = sns.heatmap(dick[:][:]['diff_ts'], vmin=0, vmax=0.25, xticklabels=np.
 ↪arange(121))
# ax.set_yticks(np.arange(len(st_df.index)))
# ax.set_yticklabels(st_df.index)
# ax.set_xticks(np.arange(121))
# ax.set_xlim(0,120)
# plt.title('HEATMAP SHOWING HOW THE SUM OF THE EXCEL & NUMPY DATASETS DIFFER␣
 ↪PER SHEET FOR EVERY HOUSE')
# ax.set_ylabel(None)
# ax.set_xlabel(None)
# plt.savefig('heetmapje_week5_v1.0.png', dpi=1200)
```

Heatmap #2 (count of ts-diff >= 1 hr)

[ ]:
```
#Select the path to the datasets and the sheets to be collected.
path = '/home/18062814/notebooks/zero/DATA/'

sheets =␣
 ↪['energyImmersion','energyHeatpump','smartMeter','energyWtwReg','solar']

#Define a function with which to compute the difference in timestep per sheet.
```

```
def add_diff(df):
    try:
        df[len(df.columns)+1] = abs(df[0].diff()-300).fillna(0)
    except:
        df[len(df.columns)+1] = 0

    return df

#Create an empty dictionary object; Key = sheetname, Value = house number.
#Start a for loop to add dictionaries to it for each house; Key = house number, Value = dataframe containing data of selected sheet.
D = {}

for i in range(1,121):
    for sheet in sheets:
        df = pd.DataFrame(np.load(path + sheet + '_' + nParse3(i) + '.npy'))
        df = add_diff(df)

        col_names = list(df.columns)
        col_names[-1] = "diff_ts"
        df.columns = col_names

        if sheet in D.keys():
            D[sheet][nParse3(i)] = df
        else:
            D[sheet] = {nParse3(i):df}

#Create an empty list; containing the mean, max and min value of diff_ts per sheet for each house.
#Loop over the data stored in dictionary.
data_list = []


for i in range(1, 121):
    for sheet in sheets:
        data = D[sheet][nParse3(i)]['diff_ts']

        row = [sheet,i,data.mean(),data.min(),data.max(), np.count_nonzero(data >= 3600)]
        data_list.append(row)

#Create a dataframe for each of the values for diff_ts in the list; MEAN, MIN, MAX, Count
data_df = pd.DataFrame(data_list,columns=["Sheet","HouseNumber","MEAN","MIN","MAX","Count"])
data_df_mean = data_df.pivot('Sheet','HouseNumber','MEAN')
```

```
data_df_min = data_df.pivot('Sheet','HouseNumber','MIN')
data_df_max = data_df.pivot('Sheet','HouseNumber','MAX')
data_df_count = data_df.pivot('Sheet','HouseNumber', 'Count')


%matplotlib notebook
plt.subplots(figsize=(30,4))
ax = sns.heatmap(data_df_count, xticklabels=range(121))
ax.set_yticks(np.arange(5))
ax.set_xticks(np.arange(121))
plt.tight_layout()
plt.title('HEATMAP SHOWING THE COUNT OF WHEN THE DIFFERENCE IN TIMESTAMP PER␣
 ↪SHEET FOR EACH HOUSE IS GREATER OR EQUAL TO 1 HOUR')
# plt.savefig('heetmappie_count_np.png', dpi=600)
```

Heatmap #3 (count of ts-diff >= 1 hr for the calmer regions in the data)

[95]:
```
#list of houses to remove
h_ext = [27, 29, 31, 32, 33, 34, 35, 36, 44, 66, 82, 101, 103, 110, 116]

#Select the path to the datasets and the sheets to be collected.
path = '/home/18062814/notebooks/zero/DATA/'

sheets =␣
 ↪['energyImmersion','energyHeatpump','smartMeter','energyWtwReg','solar']

#Define a function with which to compute the difference in timestep per sheet.
def add_diff(df):
    try:
        df[len(df.columns)+1] = abs(df[0].diff()-300).fillna(0)
    except:
        df[len(df.columns)+1] = 0
    return df

#Create an empty dictionary object; Key = sheetname, Value = house number.
#Start a for loop to add dictionaries to it for each house; Key = house number,␣
 ↪Value = dataframe containing data of selected sheet.
D = {}

for i in range(1,121):
    for sheet in sheets:
        df = pd.DataFrame(np.load(path + sheet + '_' + nParse3(i) + '.npy'))
        df = add_diff(df)

        col_names = list(df.columns)
        col_names[-1] = "diff_ts"
        df.columns = col_names
```

```
        if sheet in D.keys():
            D[sheet][nParse3(i)] = df
        else:
            D[sheet] = {nParse3(i):df}

#Create an empty list; containing the mean, max and min value of diff_ts per␣
 ↪sheet for each house.
#Loop over the data stored in dictionary.
data_list = []


for i in range(1, 121):
    if i in h_ext:
        row = ["solar",i,np.nan]
        data_list.append(row)
        continue
    for sheet in sheets:
        data = D[sheet][nParse3(i)]['diff_ts']

        row = [sheet,i,np.count_nonzero(data >= 3600)]
        data_list.append(row)

#Create a dataframe for each of the values for diff_ts in the list; MEAN, MIN,␣
 ↪MAX, Count
data_df = pd.DataFrame(data_list,columns=["Sheet","HouseNumber","Count"])
data_df_count = data_df.pivot('Sheet','HouseNumber', 'Count')

# data_df_count.columns = np.where(data_df_count.columns in h_ext,␣
 ↪data_df_count.columns, np.nan)

%matplotlib notebook
plt.subplots(figsize=(30,4))
ax = sns.heatmap(data_df_count, xticklabels=range(121), cmap='flare')
ax.set_yticks(np.arange(6))
ax.set_xticks(np.arange(121))
plt.tight_layout()
plt.title('HEATMAP SHOWING THE COUNT OF WHEN THE DIFFERENCE IN TIMESTAMP PER␣
 ↪SHEET FOR EACH HOUSE IS GREATER OR EQUAL TO 1 HOUR')
plt.savefig('heetmappie_count_np_calm.png', dpi=600)
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

[96]:
```
df.columns
```

[96]:
```
Index([0, 1, 2, 3, 'diff_ts'], dtype='object')
```

[42]:
```
data_df_count.columns
```

[42]:
```
Int64Index([  1,   2,   3,   4,   5,   6,   7,   8,   9,  10,
             …
             111, 112, 113, 114, 115, 116, 117, 118, 119, 120],
            dtype='int64', name='HouseNumber', length=120)
```

[21]:
```
np.shape(data_df_count)
```

[21]:
```
(5, 120)
```

[30]:
```
data_df_count.index
```

[30]:
```
Index(['energyHeatpump', 'energyImmersion', 'energyWtwReg', 'smartMeter',
       'solar'],
      dtype='object', name='Sheet')
```

[32]:
```
data_df_count.columns
```

[32]:
```
Int64Index([  1,   2,   3,   4,   5,   6,   7,   8,   9,  10,
             …
             111, 112, 113, 114, 115, 116, 117, 118, 119, 120],
            dtype='int64', name='HouseNumber', length=120)
```

[ ]:
```

```