EDA

April 23, 2020

1 Exploratory Data Analysis

1.1 Problem Statement

1. The folder `raw.zip` has raw files which were measured in a station. As the name indicates, there are:
   - 2 inverters,
   - 1 energy meter (named MFM) and
   - 1 meteorological substation (named WMS)
2. The raw data is a stream of data which gets recorded by the sensors on the field and is sent over the cloud.
3. The raw data is cleansed into a Gen-1 data format, here the following operations are performed:
   1. For Inverters: column i32 indicates the timestamp of the row. Make this as the first column in the Gen1 file and rename the column header to 'Timestamp').
   2. For Energy meters (MFM): Same rules as above, only difference is timestamp column is m63
   3. For Meteorological Substation (WMS): Same rules as above, only difference is timestamp column is w23

Sample Gen-1 data for some of the raw days is also provided (`\sample`)

**The data in the sample gen1 files have been bucketed into 5-min intervals. Ignore this operation*

1.2 Expected output format:

There needs to be a Gen-1 file for every raw data file. The attached `raw.zip` has data foreach substation. The output format needs to be as follows:

```
[Station ID]
     |---> [Year]
              |---->[Year-Month]
                         |--->[Substation-ID]
                                    |---> [Gen-1 Data.txt]
```

- The station ID for the given raw data is IN-023C.
- Year needs to be determined based on the timestamp of the file
- Year-Month needs to be determined based on the timestamp of the file
- Substation-ID depends on the substation read (example Inverter-1, MFM, WMS etc)
- Gen 1 `Data.txt` has the same name as the raw file.txt

Attached an example for your reference:

Files to be submitted:

- Gen-1 data (Zipped file maintaining folder structure described above)
- Python Code used to generate Gen-1 data with comments

1.3 Data Exploration

[1]:
```
# importing libraries

import os

import pandas as pd
import numpy as np
```

1.3.1 Viewing directory structure

[2]:
```
!tree -L 4 data # using the tree function in linux
```

```
data
   [IN-023C]
       2018
           2018-12
               Inverter_1
               Inverter_2
               MFM
               WMS
       2019
           2019-01
               Inverter_1
               Inverter_2
               MFM
               WMS

13 directories, 0 files
```

1.3.2 Reading a sample file

[3]:
```
# reading invertor 1 sample file

sample_read = pd.read_csv('data/[IN-023C]/2018/2018-12/Inverter_1/
↪[IN-023C]-I1-2018-12-01.txt', sep = '\t')
```

[4]:
```
sample_read
```

[4]:
```
     i1  i2    i3  i4                   i5  i6  i7  i8   i9  i10  …  i45  \
0   NaN   2  CT08   1  2018-12-01 00:00:04   0   1   3  0.0  0.0  …  0.0
```

```
1     NaN   2  CT08   1  2018-12-01 00:01:23   0   1   3  0.0  0.0  …  0.0
2     NaN   2  CT08   1  2018-12-01 00:02:43   0   1   3  0.0  0.0  …  0.0
3     NaN   2  CT08   1  2018-12-01 00:04:04   0   1   3  0.0  0.0  …  0.0
4     NaN   2  CT08   1  2018-12-01 00:06:15   0   1   3  0.0  0.0  …  0.0
…    ..  ..   …  ..                    …  ..  ..  ..  …   …   …  …
1049 NaN   2  CT08   1  2018-12-01 23:52:29   0   1   3  0.0  0.0  …  0.0
1050 NaN   2  CT08   1  2018-12-01 23:53:49   0   1   3  0.0  0.0  …  0.0
1051 NaN   2  CT08   1  2018-12-01 23:55:09   0   1   3  0.0  0.0  …  0.0
1052 NaN   2  CT08   1  2018-12-01 23:56:28   0   1   3  0.0  0.0  …  0.0
1053 NaN   2  CT08   1  2018-12-01 23:57:48   0   1   3  0.0  0.0  …  0.0

      i46  i47  i48  i49  i50  i51  i52  i53  i54
0     0.0  0.0  0.0    0    0    0    0    0   34
1     0.0  0.0  0.0    0    0    0    0    0   34
2     0.0  0.0  0.0    0    0    0    0    0   34
3     0.0  0.0  0.0    0    0    0    0    0   34
4     0.0  0.0  0.0    0    0    0    0    0   34
…    …   …   …   …   …   …   …   …   …
1049  0.0  0.0  0.0    0    0    0    0    0   35
1050  0.0  0.0  0.0    0    0    0    0    0   35
1051  0.0  0.0  0.0    0    0    0    0    0   35
1052  0.0  0.0  0.0    0    0    0    0    0   35
1053  0.0  0.0  0.0    0    0    0    0    0   35

[1054 rows x 54 columns]
```

[5]:
```python
# viewing timestamp

sample_read[['i32']].transpose()
```

[5]:
```
                       0                    1                    2     \
i32  2018-12-01 00:00:04  2018-12-01 00:01:23  2018-12-01 00:02:43

                       3                    4                    5     \
i32  2018-12-01 00:04:04  2018-12-01 00:06:15  2018-12-01 00:07:34

                       6                    7                    8     \
i32  2018-12-01 00:08:54  2018-12-01 00:10:14  2018-12-01 00:11:34

                       9     …                 1044                 1045  \
i32  2018-12-01 00:12:54  …  2018-12-01 23:44:31  2018-12-01 23:45:50

                       1046                 1047                 1048  \
i32  2018-12-01 23:47:11  2018-12-01 23:49:49  2018-12-01 23:51:10

                       1049                 1050                 1051  \
i32  2018-12-01 23:52:29  2018-12-01 23:53:49  2018-12-01 23:55:09
```

```
                   1052                 1053
i32  2018-12-01 23:56:28  2018-12-01 23:57:48

[1 rows x 1054 columns]
```

1.4 Process flow

- ☒ Construct folder structure of `/data`
- ☒ Make a `/submission` folder using the same folder structure as `/data`
- ☒ Edit column name of the timestamp column and make it the first column (Repeat for each file in all folders)
- ☒ Save file to the `/submission` folder (Repeat for each file in all folders)

[6]:
```python
def ensure_dir(dir_path):
    """
    Check if directory exists, else create it.

    Keyword arguments:
    dir_path -- directory path
    """
    directory = os.path.dirname(dir_path)
    if not os.path.exists(directory):
        print('Directory Exception: ' + dir_path + ' not available, creating␣
 ↪now...')
        os.makedirs(directory)
```

[7]:
```python
def change_timestamp(source_file_path, dest_file_path, ts_colnames):
    """
    Read file, change column name to timestamp and save it to new destination

    Keyword arguments:
    source_file_path -- file to read from
    dest_file_path -- file to read into
    ts_columns -- list of column names to update
    """
    # reading file as a dataframe
    file = pd.read_csv(source_file_path, sep = '\t')

    # changing respective column name to Timestamp
    cols = np.array(file.columns)
    cols[file.columns.isin(ts_colnames)] = 'Timestamp'
    file.columns = cols

    # moving timestamp to first column
    cols = list(cols)
    cols.insert(0, cols.pop(cols.index('Timestamp')))
    file = file[cols]

    # saving file in new directory
    file.to_csv(dest_file_path, index = False, sep = '\t', na_rep='NULL')
```

[8]:
```
def traversal_modify(source = 'data', destination = 'submission', ts_colnames =␣
 ↪['i32','m63','w23']):
    """
    Traverse the source folder, modify the column names then store resulting␣
 ↪file in destination folder

    Keyword arguments:
    source -- directory to copy from
    destination -- directory to copy to
    ts_colnames -- list of column names to update
    """
    for root, dirs, files in os.walk(source):

        # skipping all hidden files
        files = [f for f in files if not f[0] == '.']
        dirs[:] = [d for d in dirs if not d[0] == '.']

        # dirs return empty when on leaf of folder structure
        if not dirs:
            root_dest = root.replace(source, destination)

            # ensure directory exists
            ensure_dir(root_dest + '/')

            for f in files:
                # modifying file
                change_timestamp(source_file_path = root + "/" + f,␣
 ↪dest_file_path = root_dest + "/" + f, ts_colnames = ts_colnames)
```

[9]:
```
traversal_modify()
```

1.4.1 Viewing Output

[10]:
```
# reading invertor 1 sample file

sample_read = pd.read_csv('submission/[IN-023C]/2018/2018-12/Inverter_1/
 ↪[IN-023C]-I1-2018-12-01.txt', sep = '\t')
```

[11]:
```
sample_read
```

```
[11]:                    Timestamp  i1  i2    i3  i4                  i5  i6  i7  i8  \
      0     2018-12-01 00:00:04 NaN   2  CT08   1  2018-12-01 00:00:04   0   1   3
      1     2018-12-01 00:01:23 NaN   2  CT08   1  2018-12-01 00:01:23   0   1   3
      2     2018-12-01 00:02:43 NaN   2  CT08   1  2018-12-01 00:02:43   0   1   3
      3     2018-12-01 00:04:04 NaN   2  CT08   1  2018-12-01 00:04:04   0   1   3
      4     2018-12-01 00:06:15 NaN   2  CT08   1  2018-12-01 00:06:15   0   1   3
      …                       …  ..  ..   …  ..                   …  ..  ..  ..
      1049  2018-12-01 23:52:29 NaN   2  CT08   1  2018-12-01 23:52:29   0   1   3
      1050  2018-12-01 23:53:49 NaN   2  CT08   1  2018-12-01 23:53:49   0   1   3
      1051  2018-12-01 23:55:09 NaN   2  CT08   1  2018-12-01 23:55:09   0   1   3
      1052  2018-12-01 23:56:28 NaN   2  CT08   1  2018-12-01 23:56:28   0   1   3
      1053  2018-12-01 23:57:48 NaN   2  CT08   1  2018-12-01 23:57:48   0   1   3

             i9  …  i45  i46  i47  i48  i49  i50  i51  i52  i53  i54
      0     0.0  …  0.0  0.0  0.0  0.0    0    0    0    0    0   34
      1     0.0  …  0.0  0.0  0.0  0.0    0    0    0    0    0   34
      2     0.0  …  0.0  0.0  0.0  0.0    0    0    0    0    0   34
      3     0.0  …  0.0  0.0  0.0  0.0    0    0    0    0    0   34
      4     0.0  …  0.0  0.0  0.0  0.0    0    0    0    0    0   34
      …     …   …    …    …    …    …    …    …    …    …    …    …
      1049  0.0  …  0.0  0.0  0.0  0.0    0    0    0    0    0   35
      1050  0.0  …  0.0  0.0  0.0  0.0    0    0    0    0    0   35
      1051  0.0  …  0.0  0.0  0.0  0.0    0    0    0    0    0   35
      1052  0.0  …  0.0  0.0  0.0  0.0    0    0    0    0    0   35
      1053  0.0  …  0.0  0.0  0.0  0.0    0    0    0    0    0   35

      [1054 rows x 54 columns]
```

```
[12]: sample_read.columns
```

```
[12]: Index(['Timestamp', 'i1', 'i2', 'i3', 'i4', 'i5', 'i6', 'i7', 'i8', 'i9',
             'i10', 'i11', 'i12', 'i13', 'i14', 'i15', 'i16', 'i17', 'i18', 'i19',
             'i20', 'i21', 'i22', 'i23', 'i24', 'i25', 'i26', 'i27', 'i28', 'i29',
             'i30', 'i31', 'i33', 'i34', 'i35', 'i36', 'i37', 'i38', 'i39', 'i40',
             'i41', 'i42', 'i43', 'i44', 'i45', 'i46', 'i47', 'i48', 'i49', 'i50',
             'i51', 'i52', 'i53', 'i54'],
            dtype='object')
```

1.5 References

1. StackOverflow - List directory tree structure in python?
2. StackOverflow - os.walk without hidden folders
3. StackOverflow - safely create a nested directory?
4. StackOverflow - syntax for bringing a list element to the front