Data Wrangling for Beginners

August 18, 2020

This workshop is geared towards beginners in data science and AI. To begin install python 3 (download) or anacoda (download).

The following datasets would be used throughout this worksheet:

- IoT sensor dataset
- Grid bikeshare data

0.1 Content

- Dealing with columns in a dataset
- Selecting subsets and merging of datasets.
- Visualization
- Functions and Reusable code

0.2 Dealing with columns in a dataset

Manipulating data columns, making them easy to interprete and use in analysis.

[1]:
```
import pandas as pd #pip install pandas
iot=pd.read_csv('iot_telemetry_data.csv')
#iot.head(3)
```

How to get more information on the dataset?

[2]:
```
#iot.shape # number rows and columns
#iot.describe() # stats of numerical columns
#iot.info() # types, number of rows
#iot.isna().sum() # number of empty cells
```

How to get information on functions?

[3]:
```
#sum?
help('sum')
```

```
Help on built-in function sum in module builtins:

sum(iterable, start=0, /)
    Return the sum of a 'start' value (default: 0) plus an iterable of numbers

    When the iterable is empty, return the start value.
```

```
This function is intended specifically for use with numeric values and may
reject non-numeric types.
```

Handling timestamp in a dataset

[4]:
```
iot.ts=pd.to_datetime(iot['ts'], unit='s')
#iot.head(3)
```

The decimal points on the columns with type double.

[5]:
```
iot[['co','lpg','smoke','temp']]=iot[['co','lpg','smoke','temp']].apply(lambda␣
↪x:round(x,3))
#iot.head(3)
```

0.2.1 Exercise I:

We would look at the free bike dataset.

1. Import json file

[6]:
```
import json
BShare=pd.read_json('free-bike-status-1.json')
BShare=json.dumps(BShare.data[0])
BShare=pd.read_json(BShare)
#BShare.head(3)
```

2. Removing unecessary info

[7]:
```
BShare['bike_id']=BShare['bike_id'].apply(lambda x:x.split('_')[1])
#BShare.head()
```

3. Save in Excel

[8]:
```
BShare['rec_update']=iot['ts'].iloc[:21]
BShare.to_excel('bike-share.xlsx')

iot.iloc[:100].to_excel('iot.xlsx')
```

0.2.2 Challenge I:

1. Import bike-share excel file.
2. Using the iot excel dataset, modify the device column by removing the colon that separate each term (for example 1c:bf:ce:15:ec:4d becomes 1cbfce15ec4d).

0.3 Selecting subsets and merging of datasets.

Selecting the appropraite subset of a data to use as well as merging different datasets are very important.

[9]:
```
#iot.iloc[:5] # selecting rows
iot.loc[17:20,['device','co','smoke']] # select both rows and columns
#iot.loc[:2,iot.columns[3:6]] # when we only know the position of the columns
```

[9]:

	device	co	smoke
17	b8:27:eb:bf:9d:51	0.005	0.020
18	1c:bf:ce:15:ec:4d	0.004	0.019
19	b8:27:eb:bf:9d:51	0.005	0.020
20	00:0f:00:70:91:0a	0.003	0.014

From the iot_telemery csv dataset, we would select the data of device b8:27:eb:bf:9d:51.

[10]:
```
iot_d1=iot[iot.device=='b8:27:eb:bf:9d:51']
#iot_d1.head(3)
```

We can add more conditions

[11]:
```
iot_d1=iot[(iot.device=='b8:27:eb:bf:9d:51') & (iot.co>0.005)].loc[:,␣
↪['ts','device','temp','humidity']]
#iot_d1.head(3)
```

If just device b8:27:eb:bf:9d:51 is known and we dont want its information in the dataset

[12]:
```
iot_d2=iot[iot.device!='b8:27:eb:bf:9d:51'].loc[:,␣
↪['ts','device','temp','humidity']]
#iot_d2.head(3)
```

We would merge the two new datasets to reconstruct the old dataset.

[13]:
```
#iot_d1.append(iot_d2) # most used method
iot_d=pd.concat([iot_d1,iot_d2])
#iot_d.tail(3)
```

0.3.1 Exercise II:

1. Creating a vector from taking the last 30 elements in a column ts of iot_d2 and assign to first 30 elements in iot_d1, for example

$$x = [2,6,4,8,1,5,8,9], new_x = [9,8,5,1]$$

[14]:
```
iot_d1['ts'].values[:30]=iot_d2['ts'].values[-30:]
#iot_d1.head(3)
```

2.Create a new dataset of iot data merging iot_d1 and iot_d2 based on column ts.

[15]:
```
iot_d=pd.merge(iot_d1,iot_d2, on='ts')
#iot_d.head(3)
```

3. Select the iot data which was recorded between 12/07/2020 to 15/07/2020.

[16]:
```
iot_d3=iot[(iot.ts>'2020-07-12 00:00:00') & (iot.ts<'2020-07-16 00:00:00')]
#iot_d3.head(3)
```

0.3.2 Challenge II

1. Merge the new bike-share.xlsx and iot.xlsx depending on time
2. Create a new dataset which is the subset of the one in 2. where the co value is greater than the mean value of the iot.xlsx dataset.

0.4 Functions and Reusable code

It is a good practice to build function out of the codes we use frequently instead of copying and pasting all over the script.

- Function: We will use this as an example

$$x = \frac{-b \pm \sqrt{(b^2 - 4ac)}}{2a}$$

[17]:
```
import numpy as np
def Quad(a,b,c):
    if b^2-4*a*c >=0:
        return [(-b-np.sqrt(b^2-4*a*c))/(2*a),(-b+np.sqrt(b^2-4*a*c))/(2*a)]
    else:
        A=-b/(2*a)
        B= round(np.sqrt(-(b^2-4*a*c))/ (2*a),4)
    return ['complex', A, B]
```

[18]:
```
Quad(2,2,4)
```

[18]:
```
['complex', -0.5, 1.4142]
```

[19]:
```
# function to read our json file
def Read_json(js):
    jsdf=pd.read_json(js)
    jsdf=json.dumps(jsdf.data[0])
    jsdf=pd.read_json(jsdf)
    return jsdf
```

- Process: This is a function without a return parameter

[20]:
```
# Process to convert json to excel
def con_json_xlsx(js,name):
    df=Read_json(js)
    df.to_excel(name+'.xlsx')

# using the process
con_json_xlsx('free-bike-status-1.json','bike')
```

Importing another python script and calling a function from it

[21]:
```
from docfile import *
#add_4(2,5)
```