Springboard Capstone Project-CitiBike Data Wrangling

Cecilia Lee

In the following, we will describe the steps involved in wrangling with the data sets that will be used later for CitiBike analysis. The three data sets can be downloaded from the links below:

CitiBike Station Data: https://data.cityofnewyork.us/NYC-BigApps/Citi-Bike-Live-Station-Feed-JSON-/p94q-8hxh

CitiBike Trip Data: https://www.citibikenyc.com/system-data

2016 NYC Weather Data: https://www.kaggle.com/mathijs/weather-data-in-new-york-city-2016

- **Step 1:** For the CitiBike station data, it's a JSON file (stations.json). Therefore, we reviewed and picked up the information we, and saved all the cleaned information to a csv file for later use.

```
#import related library
import pandas as pd
import json
from pandas.io.json import json_normalize
# load citibike stations json file as a string, and as a Dataframe
stations_str=json.load((open('odata/stations.json')))
stations_df = pd.read_json('odata/stations.json')
# review the dataset
print(stations_df.head())
```

```
executionTime                                    stationBeanList
0  2018-01-13 07:44:42 PM  {'id': 72, 'stationName': 'W 52 St & 11 Ave', ...
1  2018-01-13 07:44:42 PM  {'id': 79, 'stationName': 'Franklin St & W Bro...
2  2018-01-13 07:44:42 PM  {'id': 82, 'stationName': 'St James Pl & Pearl...
3  2018-01-13 07:44:42 PM  {'id': 83, 'stationName': 'Atlantic Ave & Fort...
4  2018-01-13 07:44:42 PM  {'id': 116, 'stationName': 'W 17 St & 8 Ave', ...
```

```
print(stations_str)
```

{'executionTime': '2018-01-13 07:44:42 PM', 'stationBeanList': [{'id': 72, 'stationName': 'W 52 St & 11 Ave', 'availableDocks': 38, 'totalDocks': 39, 'latitude': 40.76727216, 'longitude': -73.99392888, 'statusValue': 'In Service', 'statusKey': 1, 'availableBikes': 1, 'stAddress1': 'W 52 St & 11 Ave', 'stAddress2': '

```
#we found that 'stationBeanList' has the information we need, we normalized and saved it to nstations
nstations=json_normalize(stations_str, 'stationBeanList')
print(nstations.head())
nstations.info()
# nstations is a dataframe with 815 entries and 18 columns. However, we only need few columns of information.
# choose the station that is in service
nstations=nstations[nstations.statusKey==1]
# select the columns we want and drop the rows that with no information
nstations=nstations[['id','stationName','latitude','longitude','totalDocks']]
```

```
nstations.dropna(how='any')
#check if there is any repeated station
print(nstations.id.unique().size==nstations.id.size)
# stations are unique, save the data into a csv file
nstations.to_csv('pdata/stations.csv', sep=',')
```

- **Step 2:** For the CitiBike trip data. First we tried to use JupterNotebook. However, the file is too big and the process is very slow. So, I switched to use PyCharm as IDE. Here, we picked the 2016 NYC data as the data set that will be discussed in my Capstone project.

```
# import all related packages
import pandas as pd
import os
# get all the raw data filenames (CitiBike-trips) from the data directory
file_list = os.listdir("./odata/2016/")
# grab the characters of month from file names:
def month_chars(x):
    return (x[4:5])
# sort all the filenames and put to a list
sorted(file_list, key=month_chars)
print(file_list)
# try to load each file into a dataframe, and concated all the dataframes into a big datafrme
#But, we found each file is very big, when I concated more dataframe, the system displayed out of memory error
#Therefore, we deal with two months data each time
df_list = []
for filename in file_list[0:2]:
    df_list.append(pd.read_csv('odata/2016/'+filename))
    trip2016q1_df = pd.concat(df_list)
#Review the info of the dataframe
trip2016q1_df.info()
print(trip2016q1_df.head())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1070352 entries, 0 to 560873
Data columns (total 15 columns):
tripduration              1070352 non-null int64
starttime                 1070352 non-null object
stoptime                  1070352 non-null object
start station id          1070352 non-null int64
start station name        1070352 non-null object
start station latitude    1070352 non-null float64
start station longitude   1070352 non-null float64
end station id            1070352 non-null int64
end station name          1070352 non-null object
end station latitude      1070352 non-null float64
end station longitude     1070352 non-null float64
bikeid                    1070352 non-null int64
usertype                  1070352 non-null object
```

```
birth year              1015981 non-null float64
gender                  1070352 non-null int64
dtypes: float64(5), int64(5), object(5)
memory usage: 110.2+ MB
  tripduration        starttime         stoptime  start station id  \
0          923  1/1/2016 00:00:41  1/1/2016 00:16:04              268
1          379  1/1/2016 00:00:45  1/1/2016 00:07:04              476
2          589  1/1/2016 00:00:48  1/1/2016 00:10:37              489
3          889  1/1/2016 00:01:06  1/1/2016 00:15:56              268
4         1480  1/1/2016 00:01:12  1/1/2016 00:25:52             2006
      start station name  start station latitude  start station longitude  \
0   Howard St & Centre St              40.719105              -73.999733
1        E 31 St & 3 Ave               40.743943              -73.979661
2       10 Ave & W 28 St               40.750664              -74.001768
3   Howard St & Centre St              40.719105              -73.999733
4  Central Park S & 6 Ave              40.765909              -73.976342
  end station id          end station name  end station latitude  \
0           3002  South End Ave & Liberty St             40.711512
1            498         Broadway & W 32 St             40.748549
2            284       Greenwich Ave & 8 Ave             40.739017
3           3002  South End Ave & Liberty St             40.711512
4           2006       Central Park S & 6 Ave             40.765909
  end station longitude  bikeid    usertype  birth year  gender
0           -74.015756   22285  Subscriber      1958.0       1
1           -73.988084   17827  Subscriber      1969.0       1
2           -74.002638   21997  Subscriber      1982.0       2
3           -74.015756   22794  Subscriber      1961.0       2
4           -73.976342   14562  Subscriber      1952.0       1
```

#we can see that only two month bike trip data, we have 1070352 entries and 15 columns, and the memory usuage is 110.2+ MB

#Therefore, we removed some rows with missing data and deleted some columns that we won't use

```
drop_columns=['stoptime','start station name','start station latitude','start station longitude','end station name','end station latitude','end station longitude']
trip2016q1_df.dropna(how='any')
trip2016q1_df=trip2016q1_df.drop(drop_columns,axis=1)
```

#check the data again

```
trip2016q1_df.info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1070352 entries, 0 to 560873
Data columns (total 8 columns):
tripduration       1070352 non-null int64
starttime          1070352 non-null object
start station id   1070352 non-null int64
end station id     1070352 non-null int64
bikeid             1070352 non-null int64
usertype           1070352 non-null object
birth year         1015981 non-null float64
gender             1070352 non-null int64
```

dtypes: float64(1), int64(5), object(2)
memory usage: 65.3+ MB

 # Now, it only has 8 columns now with 65 memory usage.

 # We save the cleaned data to a csv file for later use.

trip2016q1_df.to_csv('pdata/2016Q6.csv')

- **Step 3**: Here, we will review the weather data, clean it, and make sure the date format is right, and save to a new csv file

#import all related package, and load the weather data

import pandas as pd
filename="odata/2016_weather_centralpark.csv"
weather_df = pd.read_csv(filename)

#review the data

```
print(weather_df.head(20))
```

	date	maximum temperature	minimum temperature	average temperature	\
0	1/1/2016	42	34	38.0	
1	2/1/2016	40	32	36.0	
2	3/1/2016	45	35	40.0	
3	4/1/2016	36	14	25.0	
4	5/1/2016	29	11	20.0	
5	6/1/2016	41	25	33.0	
6	7/1/2016	46	31	38.5	
7	8/1/2016	46	31	38.5	
8	9/1/2016	47	40	43.5	
9	10/1/2016	59	40	49.5	
10	11/1/2016	40	26	33.0	
11	12/1/2016	44	25	34.5	
12	13-1-2016	30	22	26.0	
13	14-1-2016	38	22	30.0	
14	15-1-2016	51	34	42.5	
15	16-1-2016	52	42	47.0	
16	17-1-2016	42	30	36.0	
17	18-1-2016	31	18	24.5	
18	19-1-2016	28	16	22.0	
19	20-1-2016	37	27	32.0	

#However, we can see there is some problem in the date format

#Here, we covert the "date" column to datetime objects to solve this problem

weather_df['date']=pd.to_datetime(weather_df['date'],dayfirst=True)

#review the data again

print(weather_df.head())
print(weather_df[**'date'**])
weather_df.info()

```
0     2016-01-01
1     2016-01-02
2     2016-01-03
3     2016-01-04
4     2016-01-05
5     2016-01-06
6     2016-01-07
7     2016-01-08
8     2016-01-09
9     2016-01-10
10    2016-01-11
11    2016-01-12
12    2016-01-13
```

```
13    2016-01-14
14    2016-01-15
15    2016-01-16
16    2016-01-17
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 366 entries, 0 to 365
Data columns (total 7 columns):
date                    366 non-null datetime64[ns]
maximum temperature     366 non-null int64
minimum temperature     366 non-null int64
average temperature     366 non-null float64
precipitation           366 non-null object
snow fall               366 non-null object
snow depth              366 non-null object
dtypes: datetime64[ns](1), float64(1), int64(2), object(3)
memory usage: 15.8+ KB
```

#save the data to a csv file for later use

weather_df.to_csv('pdata/2016_weatherNYC.csv')

Files:

CitiBike-Stations Generation from JSON.ipynb
Data Wrangling-trip.py
Data Wrangling-weather.py