AbhijitMandal_DSC540_Milestone3

May 3, 2021

0.0.1 DSC 540 Week 7-8 - Milestone 3

Abhijit Mandal

0.0.2 Milestone 3

Perform at least 5 data transformation and/or cleansing steps to your website data. For example:

- Replace Headers
- Format data into a more readable format
- Identify outliers and bad data
- Find duplicates
- Fix casing or inconsistent values
- Conduct Fuzzy Matching

0.0.3 Dataset

HTML - I will be scrapping data from https://www.worldometers.info/coronavirus/#countries to get the covid details for all countries

0.0.4 Load the necessary libraries.

[310]:
```
#import libraries
import os, sys
import json
import pandas as pd
import numpy as np
from numpy import int64
import requests, io
import urllib.request
from bs4 import BeautifulSoup
import matplotlib.pyplot as plt

# Basic plotting packages
import matplotlib.pyplot as plt
# advanced ploting
import seaborn as sns

# interactive visualization
import plotly.express as px
```

```
import plotly.graph_objs as go
# import plotly.figure_factory as ff
from plotly.subplots import make_subplots
```

0.0.5 2. Reading the website data

[311]:
```
url = "https://www.worldometers.info/coronavirus/#countries"
response = requests.get(url)
class HTMLTableParser:

    def parse_url(self, url):
        response = requests.get(url)
        soup = BeautifulSoup(response.text, 'lxml')
        return [(table['id'],self.parse_html_table(table))\
                for table in soup.find_all('table')]

# HTML parser method to clean messy data
    def parse_html_table(self, table):
        n_columns = 0
        n_rows=0
        column_names = []

        # Find number of rows and columns
        # we also find the column titles if we can
        for row in table.find_all('tr'):

            # Determine the number of rows in the table
            td_tags = row.find_all('td')
            if len(td_tags) > 0:
                n_rows+=1
                if n_columns == 0:
                    # Set the number of columns for our table
                    n_columns = len(td_tags)

            # Handle column names if we find them
            th_tags = row.find_all('th')
            if len(th_tags) > 0 and len(column_names) == 0:
                for th in th_tags:
                    colData = th.get_text()
                    colData = colData.replace('/','').replace(' ','').
↪replace(',','').replace('\n','').replace(' ','').replace(' ','')
                    column_names.append(colData)

        # Safeguard on Column Titles
        if len(column_names) > 0 and len(column_names) != n_columns:
            raise Exception("Column titles do not match the number of columns")
```

```
        columns = column_names if len(column_names) > 0 else range(0,n_columns)
        df = pd.DataFrame(columns = columns,
                          index= range(0,n_rows))
        row_marker = 0
        for row in table.find_all('tr'):
            column_marker = 0
            columns = row.find_all('td')
            for column in columns:
                df.iat[row_marker,column_marker] = column.get_text()
                column_marker += 1
            if len(columns) > 0:
                row_marker += 1

        # Convert to float if possible
        for col in df:
            try:
                df[col] = df[col].astype(float)
            except ValueError:
                pass

        return df
```

[312]:
```
# Parsing Html data

hp = HTMLTableParser()
table = hp.parse_url(url)[0][1] # Grabbing the table from the tuple
table.head(10)
```

```
#
Country,Other
TotalCases
NewCases
TotalDeaths
NewDeaths
TotalRecovered
NewRecovered
ActiveCases
Serious,Critical
Tot Cases/1M pop
Deaths/1M pop
TotalTests
Tests/
1M pop

Population
Continent
1 Caseevery X ppl
```

```
1 Deathevery X ppl
1 Testevery X ppl
New Cases/1M pop
New Deaths/1M pop
Active Cases/1M pop
#
Country,Other
TotalCases
NewCases
TotalDeaths
NewDeaths
TotalRecovered
NewRecovered
ActiveCases
Serious,Critical
Tot Cases/1M pop
Deaths/1M pop
TotalTests
Tests/
1M pop

Population
Continent
1 Caseevery X ppl
1 Deathevery X ppl
1 Testevery X ppl
New Cases/1M pop
New Deaths/1M pop
Active Cases/1M pop
#
Country,Other
TotalCases
NewCases
TotalDeaths
NewDeaths
TotalRecovered
NewRecovered
ActiveCases
Serious,Critical
Tot Cases/1M pop
Deaths/1M pop
TotalTests
Tests/
1M pop

Population
Continent
1 Caseevery X ppl
```

```
1 Deathevery X ppl
1 Testevery X ppl
New Cases/1M pop
New Deaths/1M pop
Active Cases/1M pop
```

[312]:

	#	CountryOther	TotalCases	NewCases	TotalDeaths	NewDeaths	\
0		\nNorth America\n	38,441,002	+2,027	863,923	+83	
1		\nAsia\n	40,510,761	+18,192	531,115	+172	
2		\nSouth America\n	25,098,283		679,238		
3		\nEurope\n	44,780,108	+4,886	1,018,490	+187	
4		\nAfrica\n	4,611,244		122,616		
5		\nOceania\n	63,143	+16	1,202		
6		\n\n	721		15		
7		World	153,505,262	+25,121	3,216,599	+442	
8	1	USA	33,180,441		591,062		
9	2	India	19,925,604	+5,889	218,959	+14	

	TotalRecovered	NewRecovered	ActiveCases	SeriousCritical	…	TotalTests	\
0	30,169,231	+376	7,407,848	17,240	…		
1	34,762,920	+24,617	5,216,726	33,766	…		
2	22,687,904		1,731,141	27,796	…		
3	39,720,689	+38,451	4,040,929	29,259	…		
4	4,139,345		349,283	3,612	…		
5	59,498	+3	2,443	5	…		
6	706		0	0	…		
7	131,540,293	+63,447	18,748,370	111,678	…		
8	25,823,800		6,765,579	9,466	…	448,984,680	
9	16,293,003	+11,265	3,413,642	8,944	…	291,647,037	

	Tests1Mpop	Population	Continent	1CaseeveryXppl	\
0			North America		\n
1			Asia		\n
2			South America		\n
3			Europe		\n
4			Africa		\n
5			Australia/Oceania		\n
6					\n
7			All		\n
8	1,349,834	332,622,037	North America	10	
9	209,621	1,391,308,838	Asia	70	

	1DeatheveryXppl	1TesteveryXppl	NewCases1Mpop	NewDeaths1Mpop	ActiveCases1Mpop
0					
1					
2					
3					

```
4
5
6
7
8             563              1                                          20,340
9           6,354              5              4           0.01             2,454

[10 rows x 22 columns]
```

[313]:
```
#We can see a few special characters ("\n", "+") to remove in the table. Let's␣
↪check the dataframe.

#check bottom rows
table.tail(10)
```

[313]:

	#	CountryOther	TotalCases	NewCases	TotalDeaths	NewDeaths	\
228	221	Micronesia	1				
229	222	China	90,697	+11	4,636		
230		Total:	38,441,002	+2,027	863,923	+83	
231		Total:	40,510,761	+18,192	531,115	+172	
232		Total:	25,098,283		679,238		
233		Total:	44,780,108	+4,886	1,018,490	+187	
234		Total:	4,611,244		122,616		
235		Total:	63,143	+16	1,202		
236		Total:	721		15		
237		Total:	153,505,262	+25,121	3,216,599	+442	

	TotalRecovered	NewRecovered	ActiveCases	SeriousCritical	…	TotalTests	\
228	1		0		…		
229	85,738	+13	323	5	…	160,000,000	
230	30,169,231	+376	7,407,848	17,240	…		
231	34,762,920	+24,617	5,216,726	33,766	…		
232	22,687,904		1,731,141	27,796	…		
233	39,720,689	+38,451	4,040,929	29,259	…		
234	4,139,345		349,283	3,612	…		
235	59,498		2,443	5	…		
236	706		0	0	…		
237	131,540,293	+63,447	18,748,370	111,678	…		

	Tests1Mpop	Population	Continent	1CaseeveryXppl	\
228		116,027	Australia/Oceania	116,027	
229	111,163	1,439,323,776	Asia	15,870	
230			North America		
231			Asia		
232			South America		
233			Europe		
234			Africa		

```
235                              Australia/Oceania
236
237                                            All             \n

    1DeatheveryXppl 1TesteveryXppl NewCases1Mpop NewDeaths1Mpop  \
228
229         310,467              9          0.01
230
231
232
233
234
235
236
237

    ActiveCases1Mpop
228
229              0.2
230
231
232
233
234
235
236
237

[10 rows x 22 columns]
```

[ ]:

[314]:
```
# There are some extra special characters (\n..\n) in the dataframe.
# We need to remove the extra characters. We only need country data for mapping␣
 ↪in this tutorial.
# So we can drop the extra top and bottom rows that we do not need for data␣
 ↪processing.

#Drop top buttom unwanted rows
df= table.drop(table.index[[0,1,2,3,4,5,6,7]]).reset_index(drop=True)
#drop tail unwanted rows
df.drop(df.tail(8).index,inplace=True)
#drop new line '\n' charachter
df.replace(['\n'], '', regex=True, inplace=True)
df.replace([','], '', regex=True, inplace=True)
```

[315]:
```
#We need to format the table before starting mapping.
# The special characters in the dataframe can be removed using a loop as below:
# drop unwanted drop unwanted special characters using a loop
for col in df.columns[0:20]:
    df[col]=df[col].str.replace('+', '').str.replace(',', '').str.replace('N/
 ↪A', '').str.replace(' ', '').str.replace('  ', '').str.replace(' ', '')
```

[316]:
```
# All the extracted data is in text format and some column names are improper␣
 ↪for data processing.
# We need to rename some column names.

df1 = df.rename(columns={'CountryOther': 'Country_Name', 'SeriousCritical':␣
 ↪'Serious_Critical', 'Tot Cases1Mpop': 'Tot_Cases_1M_pop', 'Deaths1Mpop':␣
 ↪'Deaths_1M_pop', 'Tests1Mpop': 'Tests_1M_pop'})

df1.head()
```

[316]:

	#	Country_Name	TotalCases	NewCases	TotalDeaths	NewDeaths	TotalRecovered	NewRecovered	ActiveCases	Serious_Critical	…	TotalTests	Tests_1M_pop	Population	Continent	1CaseeveryXppl	1DeatheveryXppl	1TesteveryXppl	NewCases1Mpop	NewDeaths1Mpop	ActiveCases1Mpop
0	1	USA	33180441		591062		25823800		6765579	9466	…	448984680	1349834	332622037	NorthAmerica	10	563	1			20340
1	2	India	19925604	5889	218959	14	16293003	11265	3413642	8944	…	291647037	209621	1391308838	Asia	70	6354	5	4	0.01	2454
2	3	Brazil	14754910		407775		13278718		1068417	8318	…	43818216	204931	213819510	SouthAmerica	14	524	5			4997
3	4	France	5652247		104819		4636940		910488	5585	…	76694164	1172798	65394155	Europe	12	624	1			13923
4	5	Turkey	4875388		40844		4480381		354163	3532	…	47744338	561071	85094968	Asia	17	2083	2			4162

[5 rows x 22 columns]

[317]:
```
# However, it is still not enough for data processing. We need to check the␣
 ↪data type of each data frame column.
df1.dtypes
```

[317]:
```
#                   object
Country_Name        object
TotalCases          object
NewCases            object
TotalDeaths         object
NewDeaths           object
TotalRecovered      object
NewRecovered        object
ActiveCases         object
Serious_Critical    object
Tot Cases1Mpop      object
Deaths_1M_pop       object
TotalTests          object
Tests_1M_pop        object
Population          object
Continent           object
1CaseeveryXppl      object
1DeatheveryXppl     object
1TesteveryXppl      object
NewCases1Mpop       object
NewDeaths1Mpop      object
ActiveCases1Mpop    object
dtype: object
```

[318]:
```
# The data type of each column is object in the dataframe.
# So we need to convert some data types to appropriate data types in the data␣
 ↪frame.
# Type conversion is the conversion of object from one data type to another␣
 ↪data type.

#convert object columns in dataframe to numeric
df1.fillna(0, inplace=True)
df1.replace(np.nan, 0, inplace=True)
df1.replace(np.inf, 0, inplace=True)
for col in df1.columns[0:20]:
    df1[col] = pd.to_numeric(df1[col], errors='ignore')

df1.dtypes
```

[318]:
```
#                   int64
Country_Name       object
TotalCases          int64
NewCases          float64
```

```
TotalDeaths         float64
NewDeaths           float64
TotalRecovered        int64
NewRecovered        float64
ActiveCases           int64
Serious_Critical    float64
Tot Cases1Mpop      float64
Deaths_1M_pop       float64
TotalTests          float64
Tests_1M_pop        float64
Population          float64
Continent            object
1CaseeveryXppl      float64
1DeatheveryXppl     float64
1TesteveryXppl      float64
NewCases1Mpop       float64
NewDeaths1Mpop       object
ActiveCases1Mpop     object
dtype: object
```

[319]:
```
# replace null values with 0

worldometer_data = df1.replace('', np.nan).fillna(0)
worldometer_data.head()
```

[319]:

	#	Country_Name	TotalCases	NewCases	TotalDeaths	NewDeaths	\
0	1	USA	33180441	0.0	591062.0	0.0	
1	2	India	19925604	5889.0	218959.0	14.0	
2	3	Brazil	14754910	0.0	407775.0	0.0	
3	4	France	5652247	0.0	104819.0	0.0	
4	5	Turkey	4875388	0.0	40844.0	0.0	

	TotalRecovered	NewRecovered	ActiveCases	Serious_Critical	…	\
0	25823800	0.0	6765579	9466.0	…	
1	16293003	11265.0	3413642	8944.0	…	
2	13278718	0.0	1068417	8318.0	…	
3	4636940	0.0	910488	5585.0	…	
4	4480381	0.0	354163	3532.0	…	

	TotalTests	Tests_1M_pop	Population	Continent	1CaseeveryXppl	\
0	448984680.0	1349834.0	3.326220e+08	NorthAmerica	10.0	
1	291647037.0	209621.0	1.391309e+09	Asia	70.0	
2	43818216.0	204931.0	2.138195e+08	SouthAmerica	14.0	
3	76694164.0	1172798.0	6.539416e+07	Europe	12.0	
4	47744338.0	561071.0	8.509497e+07	Asia	17.0	

	1DeatheveryXppl	1TesteveryXppl	NewCases1Mpop	NewDeaths1Mpop	\

```
0          563.0             1.0            0.0              0
1         6354.0             5.0            4.0           0.01
2          524.0             5.0            0.0              0
3          624.0             1.0            0.0              0
4         2083.0             2.0            0.0              0

   ActiveCases1Mpop
0             20340
1              2454
2              4997
3             13923
4              4162

[5 rows x 22 columns]
```

[320]:
```
def plot_hbar_wm(col, n, min_pop=1000000, sort='descending'):
    df = worldometer_data[worldometer_data['Population']>min_pop]
    df = df.sort_values(col, ascending=True).tail(n)
    fig = px.bar(df,
                 x=col, y="Country_Name", color='Continent',
                 text=col, orientation='h', width=700,
                 color_discrete_sequence = px.colors.qualitative.Dark2)
    fig.update_layout(title=col+' (Only countries with > 1M Pop)',
                      xaxis_title="", yaxis_title="",
                      yaxis_categoryorder = 'total ascending',
                      uniformtext_minsize=8, uniformtext_mode='hide')
    fig.show()
```

[321]:
```
plot_hbar_wm('Tests_1M_pop', 15, 1000000)
```

[ ]:
```

```