AbhijitMandal_DSC540_Milestone2

April 23, 2021

0.0.1 DSC 540 Week 5-6 - Milestone 2

Abhijit Mandal

0.0.2 Milestone 2

Perform at least 5 data transformation and/or cleansing steps to your flat file 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

CSV – The Covid 19 data is scrapped from John Hopkins University github repo : https://github.com/CSSEGISandData/COVID-19/tree/master/csse_covid_19_data/csse_covid_19_time_series , this has

- Daily time series summary tables, including confirmed, deaths and recovered. All data is read in from the daily case report. The time series tables are subject to be updated if inaccuracies are identified in our historical data.
- Two time series tables are for the US confirmed cases and deaths, reported at the county level.
- Three time series tables are for the global confirmed cases, recovered cases and deaths. Australia, Canada and China are reported at the province/state level.
- Data is updated at a daily basis.

0.0.4 Load the necessary libraries.

[29]:
```
# import libraries
# for date and time opeations
from datetime import datetime, timedelta
# for file and folder operations
import os
# for regular expression operations
import re
# for listing files in a folder
```

```
import glob
# for getting web contents
import requests
# storing and analysing data
import pandas as pd
# for scraping web contents
from bs4 import BeautifulSoup
# numerical analysis
import numpy as np
```

[30]:
```
# Read dataset
conf_df = pd.read_csv('../time_series_covid19_confirmed_global.csv')
deaths_df = pd.read_csv('../time_series_covid19_deaths_global.csv')
recv_df = pd.read_csv('../time_series_covid19_recovered_global.csv')
```

[31]:
```
#View the data for any cleanup
conf_df.head()
```

[31]:

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	\
0	NaN	Afghanistan	33.93911	67.709953	0	0	
1	NaN	Albania	41.15330	20.168300	0	0	
2	NaN	Algeria	28.03390	1.659600	0	0	
3	NaN	Andorra	42.50630	1.521800	0	0	
4	NaN	Angola	-11.20270	17.873900	0	0	

	1/24/20	1/25/20	1/26/20	1/27/20	…	4/13/21	4/14/21	4/15/21	\
0	0	0	0	0	…	57364	57492	57534	
1	0	0	0	0	…	128752	128959	129128	
2	0	0	0	0	…	118799	118975	119142	
3	0	0	0	0	…	12614	12641	12641	
4	0	0	0	0	…	23697	23841	23951	

	4/16/21	4/17/21	4/18/21	4/19/21	4/20/21	4/21/21	4/22/21
0	57612	57721	57793	57898	58037	58214	58312
1	129307	129456	129594	129694	129842	129980	130114
2	119323	119486	119642	119805	119992	120174	120363
3	12712	12771	12805	12805	12874	12917	12942
4	24122	24300	24389	24518	24661	24883	25051

[5 rows x 461 columns]

[32]:
```
deaths_df.head()
```

[32]:

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	\
0	NaN	Afghanistan	33.93911	67.709953	0	0	
1	NaN	Albania	41.15330	20.168300	0	0	
2	NaN	Algeria	28.03390	1.659600	0	0	

```
3            NaN        Andorra  42.50630   1.521800        0        0
4            NaN         Angola -11.20270  17.873900        0        0

   1/24/20  1/25/20  1/26/20  1/27/20  …  4/13/21  4/14/21  4/15/21  \
0        0        0        0        0  …     2529     2532     2533
1        0        0        0        0  …     2326     2331     2335
2        0        0        0        0  …     3137     3141     3144
3        0        0        0        0  …      121      121      121
4        0        0        0        0  …      554      557      557

   4/16/21  4/17/21  4/18/21  4/19/21  4/20/21  4/21/21  4/22/21
0     2535     2539     2539     2546     2549     2557     2561
1     2337     2340     2342     2347     2353     2358     2364
2     3148     3152     3155     3160     3165     3172     3181
3      123      123      123      123      123      123      123
4      560      561      561      563      565      570      572

[5 rows x 461 columns]
```

[33]:
```
recv_df.head()
```

[33]:
```
  Province/State Country/Region       Lat       Long  1/22/20  1/23/20  \
0            NaN    Afghanistan  33.93911  67.709953        0        0
1            NaN        Albania  41.15330  20.168300        0        0
2            NaN        Algeria  28.03390   1.659600        0        0
3            NaN        Andorra  42.50630   1.521800        0        0
4            NaN         Angola -11.20270  17.873900        0        0

   1/24/20  1/25/20  1/26/20  1/27/20  …  4/13/21  4/14/21  4/15/21  \
0        0        0        0        0  …    52013    52022    52083
1        0        0        0        0  …    98903    99441   100013
2        0        0        0        0  …    82813    82929    83048
3        0        0        0        0  …    11932    11989    11989
4        0        0        0        0  …    22115    22144    22175

   4/16/21  4/17/21  4/18/21  4/19/21  4/20/21  4/21/21  4/22/21
0    52105    52116    52168    52244    52272    52301    52348
1   100600   101142   101584   102171   102601   103066   103582
2    83169    83286    83397    83514    83636    83765    83900
3    12105    12159    12203    12203    12285    12334    12375
4    22203    22576    22597    22600    22647    22882    22901

[5 rows x 461 columns]
```

[34]:
```
# Merge the datasets
# extract dates
dates = conf_df.columns[4:]
```

```
# melt dataframes into longer format
conf_df_long = conf_df.melt(id_vars=['Province/State', 'Country/Region', 'Lat',␣
 ↪'Long'],
                            value_vars=dates, var_name='Date',␣
 ↪value_name='Confirmed')

deaths_df_long = deaths_df.melt(id_vars=['Province/State', 'Country/Region',␣
 ↪'Lat', 'Long'],
                            value_vars=dates, var_name='Date',␣
 ↪value_name='Deaths')

recv_df_long = recv_df.melt(id_vars=['Province/State', 'Country/Region', 'Lat',␣
 ↪'Long'],
                            value_vars=dates, var_name='Date',␣
 ↪value_name='Recovered')



print(conf_df_long.shape)
print(deaths_df_long.shape)
print(recv_df_long.shape)
```

```
(125675, 6)
(125675, 6)
(118820, 6)
```

[35]:
```
# merge dataframes to get a full dataframe, we will then perform a cleanup on␣
 ↪the final dataset

full_table = pd.merge(left=conf_df_long, right=deaths_df_long, how='left',
                      on=['Province/State', 'Country/Region', 'Date', 'Lat',␣
 ↪'Long'])
full_table = pd.merge(left=full_table, right=recv_df_long, how='left',
                      on=['Province/State', 'Country/Region', 'Date', 'Lat',␣
 ↪'Long'])

full_table.head()
```

[35]:

	Province/State	Country/Region	Lat	Long	Date	Confirmed	\
0	NaN	Afghanistan	33.93911	67.709953	1/22/20	0	
1	NaN	Albania	41.15330	20.168300	1/22/20	0	
2	NaN	Algeria	28.03390	1.659600	1/22/20	0	
3	NaN	Andorra	42.50630	1.521800	1/22/20	0	
4	NaN	Angola	-11.20270	17.873900	1/22/20	0	

```
   Deaths  Recovered
0       0        0.0
1       0        0.0
2       0        0.0
3       0        0.0
4       0        0.0
```

0.0.5 1. Cleanup Activities

[36]:
```
# 1. Convert to proper date format
full_table['Date'] = pd.to_datetime(full_table['Date'])

# 2. fill na with 0
full_table['Recovered'] = full_table['Recovered'].fillna(0)

# 3. convert to int datatype
full_table['Recovered'] = full_table['Recovered'].astype('int')

# 4. fixing Country names

# 4.1 renaming countries, regions, provinces
full_table['Country/Region'] = full_table['Country/Region'].replace('Korea,␣
↪South', 'South Korea')

# 4.2 Greenland
full_table.loc[full_table['Province/State']=='Greenland', 'Country/Region'] =␣
↪'Greenland'

# 4.3 Mainland china to China
full_table['Country/Region'] = full_table['Country/Region'].replace('Mainland␣
↪China', 'China')


# 5. Removing county wise data to avoid double counting
full_table = full_table[full_table['Province/State'].str.contains(',')!=True]
```

0.0.6 2. Adding Calculated values and 3. filling missing values

[37]:
```
# Active Case = confirmed - deaths - recovered
full_table['Active'] = full_table['Confirmed'] - full_table['Deaths'] -␣
↪full_table['Recovered']

# filling missing values

# fill missing province/state value with ''
full_table[['Province/State']] = full_table[['Province/State']].fillna('')
```

```
# fill missing numerical values with 0
cols = ['Confirmed', 'Deaths', 'Recovered', 'Active']
full_table[cols] = full_table[cols].fillna(0)

# fixing datatypes
full_table['Recovered'] = full_table['Recovered'].astype(int)

# Viewing sample rows
full_table.sample(6)
```

[37]:

	Province/State	Country/Region	Lat	Long	\
19380		France	46.227600	2.213700	
47503		Oman	21.512583	55.923255	
101013	Unknown	China	NaN	NaN	
68215	Western Australia	Australia	-31.950500	115.860500	
64120	Newfoundland and Labrador	Canada	53.135500	-57.660400	
49174		Serbia	44.016500	21.005900	

	Date	Confirmed	Deaths	Recovered	Active
19380	2020-04-01	56362	4767	10934	40661
47503	2020-07-12	56015	257	36098	19660
101013	2021-01-23	0	0	0	0
68215	2020-09-26	676	9	651	16
64120	2020-09-11	270	3	0	267
49174	2020-07-18	20498	461	15179	4858

[38]:
```
# function to change value of a column in dataframe
def change_val(date, ref_col, val_col, dtnry):
    for key, val in dtnry.items():
        full_table.loc[(full_table['Date']==date) & (full_table[ref_col]==key),␣
↪val_col] = val
```

0.0.7 4. Updating bad data

[39]:
```
# we found that hubei province in China has incorrect data,
# lets see what it is and will update it with correct one
# checking values
full_table[(full_table['Date']=='2/12/20') & (full_table['Province/
↪State']=='Hubei')]
```

[39]:

	Province/State	Country/Region	Lat	Long	Date	Confirmed	\
5846	Hubei	China	30.9756	112.2707	2020-02-12	33366	

	Deaths	Recovered	Active
5846	1068	2686	29612

[40]:
```
# The confirmed deaths need to be updated to 34874 as per the latest info, we␣
↪will do that update
feb_12_conf = {'Hubei' : 34874}
change_val('2/12/20', 'Province/State', 'Confirmed', feb_12_conf)
```

[41]:
```
# Lets check the values after the update
full_table[(full_table['Date']=='2/12/20') & (full_table['Province/
↪State']=='Hubei')]
```

[41]:

	Province/State	Country/Region	Lat	Long	Date	Confirmed	Deaths	Recovered	Active
5846	Hubei	China	30.9756	112.2707	2020-02-12	34874	1068	2686	29612

0.0.8 5. Removing Outliers

[42]:
```
# there is ship rows info which contains ships with Covid-19 reported cases
# this is an outlier for our analysis so we will remove that info from our␣
↪dataframe

# ship rows containing ships with COVID-19 reported cases
ship_rows = full_table['Province/State'].str.contains('Grand Princess') | \
            full_table['Province/State'].str.contains('Diamond Princess') | \
            full_table['Country/Region'].str.contains('Diamond Princess') | \
            full_table['Country/Region'].str.contains('MS Zaandam')

# ship
ship = full_table[ship_rows]

# Latest cases from the ships
ship_latest = ship[ship['Date']==max(ship['Date'])]
# ship_latest.style.background_gradient(cmap='Pastel1_r')

# skipping rows with ships info
full_table = full_table[~(ship_rows)]
```

[43]:
```
full_table
```

[43]:

	Province/State	Country/Region	Lat	Long	Date
0		Afghanistan	33.939110	67.709953	2020-01-22
1		Albania	41.153300	20.168300	2020-01-22
2		Algeria	28.033900	1.659600	2020-01-22
3		Andorra	42.506300	1.521800	2020-01-22
4		Angola	-11.202700	17.873900	2020-01-22
…	…	…	…	…	…
125670		Vietnam	14.058324	108.277199	2021-04-22

```
125671                West Bank and Gaza  31.952200   35.233200 2021-04-22
125672                             Yemen  15.552727   48.516388 2021-04-22
125673                            Zambia -13.133897   27.849332 2021-04-22
125674                          Zimbabwe -19.015438   29.154857 2021-04-22

        Confirmed  Deaths  Recovered  Active
0               0       0          0       0
1               0       0          0       0
2               0       0          0       0
3               0       0          0       0
4               0       0          0       0
…             …     …        …     …
125670       2824      35       2490     299
125671     287680    3115     256559   28006
125672       6020    1157       2393    2470
125673      91189    1240      89117     832
125674      38018    1555      35073    1390

[122933 rows x 9 columns]
```

[ ]: