

# DTSC670: Foundations of Machine Learning Models

## Module 1

### Assignment 2: COVID-19 Data Wrangling

Name: Charles Young

The purpose of this assignment is to hone your data wrangling skills. Your task for this assignment is to perform the data preparation as instructed in the DTSC670\_Assignment\_2 pdf listed in Brightspace. After performing all the data preparation tasks outlined in the document, run the code in the "Prepare DataFrames for Grading" section.

You are supplied an Excel file called `BrazilCOVIDData.xlsx` - be sure to put the data file in the same directory as this Jupyter Notebook. *Please note that it may take around 5 minutes to read-in all of the data in this file.*

```
In [3]: #Importing Libraries here
import pandas as pd
import numpy as np
```

Reading all sheets which are present in excel file

```
In [4]: xls = pd.ExcelFile("BrazilCOVIDData.xlsx")
df_brazil_covid_data = pd.read_excel(xls, 'Brazil Covid-19 data')
df_city_area = pd.read_excel(xls, 'City area')
df_Brazil_state_stats = pd.read_excel(xls, 'Brazil State Stats', header = [4,5,6])
df_temp_by_state = pd.read_excel(xls, 'Temperature by State')
#I need a Mac
```

converting each dataframe into our requirement

## df\_brazil\_covid\_data

```
In [5]: df_brazil_covid_data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 846770 entries, 0 to 846769
Data columns (total 17 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Region                846770 non-null object
1   State                 846588 non-null object
2   Municipality          838503 non-null object
3   State-code            846770 non-null int64
4   Municipality-code     841674 non-null float64
5   Health-region-code    838503 non-null float64
```

```

6 Health-region-name      838503 non-null object
7 Date                    846770 non-null datetime64[ns]
8 Week #                  846770 non-null int64
9 Population as of 2019   843599 non-null object
10 Accumulated cases      846770 non-null int64
11 New cases              846770 non-null int64
12 Accumulated deaths     846770 non-null int64
13 New deaths             846770 non-null int64
14 New Recoveries         128 non-null float64
15 New followups (?)      128 non-null float64
16 Interior/Metropolitan  838503 non-null float64
dtypes: datetime64[ns](1), float64(5), int64(6), object(5)
memory usage: 109.8+ MB

```

dropping unnecessary columns of df\_brazil\_covid\_data

```

In [6]: df_brazil_covid_data.drop(["Municipality-code", "State-code", "Health-region-code", "Health-region-name", "Accumulated deaths", "New deaths", "New Recoveries", "New followups", "New followups (?)", "Interior/Metropolitan"], axis = 1, inplace = True)

```

Getting only the States of Brazil

```

In [7]: a = ["Rio Branco", "Maceió", "Macapá", "Manaus", "Salvador", "Fortaleza", "Brasília", "Vitória", "São Luís", "Cuiabá", "Campo Grande", "Belo Horizonte", "Belém", "João Pessoa", "Curitiba", "Rio de Janeiro", "Natal", "Porto Alegre", "Porto Velho", "Boa Vista", "Florianópolis"]
df_brazil_covid_data = df_brazil_covid_data[df_brazil_covid_data["Municipality"].isin(a)]

```

```

In [8]: df_brazil_covid_data.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 4983 entries, 7663 to 846769
Data columns (total 7 columns):
#   Column              Non-Null Count  Dtype
---  -
0   Region              4983 non-null   object
1   State               4983 non-null   object
2   Municipality        4983 non-null   object
3   Date                4983 non-null   datetime64[ns]
4   Week #              4983 non-null   int64
5   Population as of 2019 4983 non-null   object
6   Accumulated cases    4983 non-null   int64
dtypes: datetime64[ns](1), int64(2), object(4)
memory usage: 311.4+ KB

```

creating a column in which the data of covid when outbreak happens

```

In [9]: #function in pandas for applying condition and counting the days since covid outbreak
def Days_Covid_Started(week):
    global count
    global week_35
    if(week == 35):
        if(week_35 == 1):
            temp = count+1
            count = 0
            week_35 = 0
            return temp
        week_35 = week_35 + 1
        count = count + 1
    return count

```

```

else:
    count = count +1
    return count

count = 0
week_35 = 0
df_brazil_covid_data["Days_Covid_Started"] = df_brazil_covid_data.apply(lambda row : Da

```

```

In [10]: df_brazil_covid_data.reset_index(drop =True).head()

```

```

Out[10]:

```

	Region	State	Municipality	Date	Week #	Population as of 2019	Accumulated cases	Days_Covid_Started
0	Norte	RO	Porto Velho	2020-03-27	13	529544	0	1
1	Norte	RO	Porto Velho	2020-03-28	13	529544	5	2
2	Norte	RO	Porto Velho	2020-03-29	14	529544	5	3
3	Norte	RO	Porto Velho	2020-03-30	14	529544	5	4
4	Norte	RO	Porto Velho	2020-03-31	14	529544	6	5

## df\_Brazil\_state\_stats

setting the coloumns name

```

In [11]: df_Brazil_state_stats.columns = ["STATE_ABBR", "State", "Capitol City", "Region", "Size", "P
, "Urban/Rural %Pop", "No. of MD", "Per Capita GNP(R$)", "
df_Brazil_state_stats.head()

```

```

Out[11]:

```

	STATE_ABBR	State	Capitol City	Region	Size	Population	Urban/Rural %Pop	No. of MD	Per Capita GNP(R\$)	Expe
0	AC	Acre	Rio Branco	North	152581	664000	69.6/30.4	24	R\$5,413	
1	AL	Alagoas	Maceió	Northeast	27767	3557000	67.4/32.6	102	R\$3,876	
2	AP	Amapá	Macapá	North	142814	619000	93.7/6.3	16	R\$6,796	
3	AM	Amazonas	Manaus	North	1570745	3351000	77.6/22.4	62	R\$11,434	
4	BA	Bahia	Salvador	Northeast	564692	13974000	67.4/32.6	417	R\$6,350	

Getting only the State\_abbr, state and population

```

In [12]: df_Brazil_state_stats = df_Brazil_state_stats.drop(labels=['Region', 'Size',
'Urban/Rural %Pop', 'No. of MD', 'Per Capita GNP(R$)',

```

```
'Life Expentancy'],axis =1)
```

```
In [13]: df_Brazil_state_stats.head()
```

```
Out[13]:
```

	STATE_ABBR	State	Capitol City	Population
0	AC	Acre	Rio Branco	664000
1	AL	Alagoas	Maceió	3557000
2	AP	Amapá	Macapá	619000
3	AM	Amazonas	Manaus	3351000
4	BA	Bahia	Salvador	13974000

## df\_temp\_by\_state

Getting only Data of Capitals

```
In [14]: df_temp_by_state = df_temp_by_state.dropna(subset=['IS_CAPITOL'])
df_temp_by_state = df_temp_by_state.reset_index(drop =True)
#df_temp_by_state = df_temp_by_state.drop("index",axis =1)
df_temp_by_state = df_temp_by_state.drop(labels=["STATE", "IS_CAPITOL",'JAN', 'FEB',
'MAR', 'APR', 'MAY', 'JUN', 'JUL', 'AUG', 'SEP', 'OCT', 'NOV', 'DEC',
'YEARS', '# CITIES'], axis=1)
```

```
In [15]: df_temp_by_state.head(20)
```

```
Out[15]:
```

	STATE_ABBR	CITY	ANNUAL
0	AC	Rio Branco	76.6
1	AL	Maceió	76.6
2	AP	Macapá	79.9
3	AM	Manaus	81.0
4	BA	Salvador	77.5
5	CE	Fortaleza	79.9
6	DF	Brasília	69.1
7	ES	Vitória	75.6
8	GO	Goiânia	73.8
9	MA	São Luís	79.0
10	MT	Cuiabá	80.0
11	MS	Campo Grande	73.0
12	MG	Belo Horizonte	70.0
13	PA	Belém	78.6

	STATE_ABBR	CITY	ANNUAL
14	PB	João Pessoa	79.0
15	PR	Curitiba	62.2
16	PE	Recife	78.1
17	PI	Teresina	82.0
18	RJ	Rio de Janeiro	75.6
19	RN	Natal	80.0

```
In [16]: df_temp_by_state.columns
```

```
Out[16]: Index(['STATE_ABBR', 'CITY', 'ANNUAL'], dtype='object')
```

## Now Preparing Features Dataframe

```
In [17]: features = df_brazil_covid_data[["State", "Municipality", "Days_Covid_Started"]].copy()
features = features.reset_index(drop = True)
```

```
In [18]: features.rename(columns = {"State": "STATE_ABBR", "Municipality": "Capital", "Days_Covid_St
features.head()
```

```
Out[18]:
```

	STATE_ABBR	Capital	days
0	RO	Porto Velho	1
1	RO	Porto Velho	2
2	RO	Porto Velho	3
3	RO	Porto Velho	4
4	RO	Porto Velho	5

Getting Temperature value for each State and Capital From df\_temp\_by\_state

Getting Population value from each State and Capital From df\_Brazil\_state\_stats

Getting SQ\_KM of the every city from df\_city\_area

```
In [19]: features = pd.merge(features, df_temp_by_state, how="inner", on="STATE_ABBR")
features = pd.merge(features, df_Brazil_state_stats, how = "inner", on = "STATE_ABBR")
features = pd.merge(features, df_city_area, how = "inner", left_on = "STATE_ABBR", right_on
```

Dropping unnecessary coloumns

```
In [20]: features = features.drop(labels = ['CITY', 'State', 'Capitol City', 'ST', 'City'], axis =
features.head()
```

```
Out[20]:
```

	STATE_ABBR	Capital	days	ANNUAL	Population	SQ_KM
0	RO	Porto Velho	1	78.1	1567000	34091.0
1	RO	Porto Velho	2	78.1	1567000	34091.0
2	RO	Porto Velho	3	78.1	1567000	34091.0
3	RO	Porto Velho	4	78.1	1567000	34091.0
4	RO	Porto Velho	5	78.1	1567000	34091.0

Getting pop\_dense

```
In [21]: features['pop_dense'] = features['Population']/features['SQ_KM']
features.head()
```

```
Out[21]:
```

	STATE_ABBR	Capital	days	ANNUAL	Population	SQ_KM	pop_dense
0	RO	Porto Velho	1	78.1	1567000	34091.0	45.965211
1	RO	Porto Velho	2	78.1	1567000	34091.0	45.965211
2	RO	Porto Velho	3	78.1	1567000	34091.0	45.965211
3	RO	Porto Velho	4	78.1	1567000	34091.0	45.965211
4	RO	Porto Velho	5	78.1	1567000	34091.0	45.965211

Getting Sqaures of days and pop\_dense and renaming and rearranging coloumns

```
In [22]: features['days_sq']=np.sqrt((features['days']))
features['days_cube']=np.power((features['days']),3)
features['pop_dense_sq']=np.sqrt((features['pop_dense']))
features.rename(columns = {'ANNUAL':'temp', 'Population':'pop'}, inplace = True)
```

```
In [23]: features = features.drop(labels = ['STATE_ABBR', 'Capital', 'SQ_KM'],axis =1)
```

```
In [24]: features = features[['days_cube','days_sq','days','temp','pop_dense_sq','pop_dense','pop']
features.head()
```

```
Out[24]:
```

	days_cube	days_sq	days	temp	pop_dense_sq	pop_dense	pop
0	1	1.000000	1	78.1	6.779765	45.965211	1567000
1	8	1.414214	2	78.1	6.779765	45.965211	1567000
2	27	1.732051	3	78.1	6.779765	45.965211	1567000
3	64	2.000000	4	78.1	6.779765	45.965211	1567000
4	125	2.236068	5	78.1	6.779765	45.965211	1567000

```
In [25]: # Now Preparing Features Dataframe
```

```
In [26]: # Get Final Response DataFrame
response = df_brazil_covid_data
response.head()
```

```
Out[26]:
```

	Region	State	Municipality	Date	Week #	Population as of 2019	Accumulated cases	Days_Covid_Started
<b>7663</b>	Norte	RO	Porto Velho	2020-03-27	13	529544	0	1
<b>7664</b>	Norte	RO	Porto Velho	2020-03-28	13	529544	5	2
<b>7665</b>	Norte	RO	Porto Velho	2020-03-29	14	529544	5	3
<b>7666</b>	Norte	RO	Porto Velho	2020-03-30	14	529544	5	4
<b>7667</b>	Norte	RO	Porto Velho	2020-03-31	14	529544	6	5

Dropping unnecessary columns

```
In [27]: response = response.drop(labels = ['Region', 'State', 'Date', 'Week #', 'Population as o
response = response.reset_index(drop = True)
response.head()
```

```
Out[27]:
```

	Municipality	Accumulated cases	Days_Covid_Started
<b>0</b>	Porto Velho	0	1
<b>1</b>	Porto Velho	5	2
<b>2</b>	Porto Velho	5	3
<b>3</b>	Porto Velho	5	4
<b>4</b>	Porto Velho	6	5

```
In [28]: response.rename(columns = {'Municipality': 'Capital'}, inplace = True)
response.head()
```

```
Out[28]:
```

	Capital	Accumulated cases	Days_Covid_Started
<b>0</b>	Porto Velho	0	1
<b>1</b>	Porto Velho	5	2
<b>2</b>	Porto Velho	5	3
<b>3</b>	Porto Velho	5	4
<b>4</b>	Porto Velho	6	5

## Prepare DataFrames for Grading

## Do not make changes to the below code

After completing all data preparation tasks, run the following four cells to prepare your DataFrame for grading by:

1. Outputting the `features` and `response` DataFrames (you do not need to print).
2. Using the NumPy `round()` function to round all the values in both DataFrames to ***ZERO decimal places***. You are calling these `features_round` and `response_round`, respectively.
3. Computing the sum of every column for both `features_round` and `response_round`, and saving those values as `features_final` and `response_final`.

***Finally, you are printing your final answer using the `print()` function.***

***Be sure to run all cells of your notebook prior to submitting, so that all output is rendered, visible and there are no error messages.***

In [29]: `features`

Out[29]:

	days_cube	days_sq	days	temp	pop_dense_sq	pop_dense	pop
0	1	1.000000	1	78.1	6.779765	45.965211	1567000
1	8	1.414214	2	78.1	6.779765	45.965211	1567000
2	27	1.732051	3	78.1	6.779765	45.965211	1567000
3	64	2.000000	4	78.1	6.779765	45.965211	1567000
4	125	2.236068	5	78.1	6.779765	45.965211	1567000
...	...	...	...	...	...	...	...
4978	3176523	12.124356	147	69.1	20.308717	412.443985	2393000
4979	3241792	12.165525	148	69.1	20.308717	412.443985	2393000
4980	3307949	12.206556	149	69.1	20.308717	412.443985	2393000
4981	3375000	12.247449	150	69.1	20.308717	412.443985	2393000
4982	3442951	12.288206	151	69.1	20.308717	412.443985	2393000

4983 rows × 7 columns

In [30]: `response`

Out[30]:

	Capital	Accumulated cases	Days_Covid_Started
0	Porto Velho	0	1
1	Porto Velho	5	2
2	Porto Velho	5	3
3	Porto Velho	5	4



	Capital	Accumulated cases	Days_Covid_Started
4	Porto Velho	6	5
...	...	...	...
4978	Brasília	143759	147
4979	Brasília	145452	148
4980	Brasília	147127	149
4981	Brasília	148998	150
4982	Brasília	150519	151

4983 rows × 3 columns

```
In [35]: features_round = np.around(features, decimals=0)
features_final = features_round.sum(axis=0)
print(features_final)
```

```
days_cube      4.346053e+09
days_sq        4.092000e+04
days           3.787080e+05
temp            3.782550e+05
pop_dense_sq    4.754990e+05
pop_dense       6.573362e+07
pop             3.261953e+10
dtype: float64
```

```
In [36]: response_round = np.around(response, decimals=0)
response_final = response_round.sum(axis=0)
print(response_final)
```

```
Capital          Porto VelhoPorto VelhoPorto VelhoPorto VelhoPo...
Accumulated cases          59586169
Days_Covid_Started          378708
dtype: object
```