In [129]:
```
import numpy as np
import pandas as pd
import os
```

In [140]:
```
# Get COVID-19 Data from John Hopkins CSSE
df_confirmed = pd.read_csv('https://raw.githubusercontent.com/CSSEGISar
df_deaths = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandDa
```

In [215]:
```
df_confirmed.columns
```

```
Index(['Province/State', 'Country/Region', 'Lat', 'Long', '1/22/20', '1/23/20',
       '1/24/20', '1/25/20', '1/26/20', '1/27/20', '1/28/20', '1/29/20',
       '1/30/20', '1/31/20', '2/1/20', '2/2/20', '2/3/20', '2/4/20', '2/5/20',
       '2/6/20', '2/7/20', '2/8/20', '2/9/20', '2/10/20', '2/11/20', '2/12/20',
       '2/13/20', '2/14/20', '2/15/20', '2/16/20', '2/17/20', '2/18/20',
       '2/19/20', '2/20/20', '2/21/20', '2/22/20', '2/23/20', '2/24/20',
       '2/25/20', '2/26/20', '2/27/20', '2/28/20', '2/29/20', '3/1/20',
       '3/2/20', '3/3/20', '3/4/20', '3/5/20', '3/6/20', '3/7/20', '3/8/20',
       '3/9/20', '3/10/20', '3/11/20', '3/12/20', '3/13/20', '3/14/20',
       '3/15/20', '3/16/20', '3/17/20', '3/18/20', '3/19/20', '3/20/20',
       '3/21/20', '3/22/20', '3/23/20', '3/24/20', '3/25/20', '3/26/20',
       '3/27/20', '3/28/20', '3/29/20', '3/30/20', '3/31/20'],
      dtype='object')
```

In [207]:
```
# To Separate the training data sets 1/22/2020 - 3/18/2020
dates_train = list(df_confirmed.columns[4:61])
confirmed_train = pd.DataFrame(np.array(df_confirmed[df_confirmed.colum
deaths_train = pd.DataFrame(np.array(df_deaths[df_deaths.columns[4:61]]
p_before_train = list(df_confirmed['Province/State'])
p_train = pd.DataFrame([val for val in p_before_train for i in range(le
c_before_train = list(df_confirmed['Country/Region'])
c_train = pd.DataFrame([val for val in c_before_train for i in range(le
d_train = pd.DataFrame(dates_train*len(df_confirmed['Province/State']))
train = pd.concat([p_train,c_train,d_train,confirmed_train,deaths_train
train.columns = ['Province_State', 'Country_Region','Date','ConfirmedCa
train
test.to_csv('train.csv')
```

In [202]:

```
# To Separate the test data sets 3/18/2020 - 3/31/2020
dates_test = list(df_confirmed.columns[61:])
confirmed_test = pd.DataFrame(np.array(df_confirmed[df_confirmed.columr
deaths_test = pd.DataFrame(np.array(df_deaths[df_deaths.columns[61:]]).
p_before_test = list(df_confirmed['Province/State'])
p_test = pd.DataFrame([val for val in p_before_test for i in range(len(
c_before_test = list(df_confirmed['Country/Region'])
c_test = pd.DataFrame([val for val in c_before_test for i in range(len(
c_train = [val for val in c_before_train for i in range(len(dates_test)
d_test = pd.DataFrame(dates_test*len(df_confirmed['Province/State']))
test = pd.concat([p_test,c_test,d_test,confirmed_test,deaths_test],axis
test.columns = ['Province_State', 'Country_Region','Date','ConfirmedCas
test.to_csv('test.csv')
```

In [208]:

```
# Load Covid Data
df = pd.read_csv('train.csv', sep=',')
df['Date'] = pd.to_datetime(df['Date'])
train_last_date = df.Date.unique()[-1]
print(f"Dataset has training data untill : {train_last_date}")
```

```
Dataset has training data untill : 2020-03-31T00:00:00.000000000
```

In [209]:

```
wpop = pd.read_csv('WPP2019_PopulationByAgeSex_Medium.csv')
country_mapper = {
'Iran (Islamic Republic of)' : "Iran",
'Bolivia (Plurinational State of)' : 'Bolivia',
'Brunei Darussalam' : 'Brunei',
'Congo' : 'Congo (Kinshasa)',
'Democratic Republic of the Congo' : "Congo (Brazzaville)",
"Côte d'Ivoire": "Cote d'Ivoire",
"Gambia" : "Gambia, The",
"Republic of Korea": "Korea, South",
"Republic of Moldova": "Moldova",
'Réunion' : "Reunion",
'Russian Federation' : "Russia",
'China, Taiwan Province of China' : "Taiwan*",
"United Republic of Tanzania": "Tanzania",
"Bahamas": "The Bahamas",
"Gambia": "The Gambia",
"United States of America (and dependencies)" : "US",
"Venezuela (Bolivarian Republic of)" : "Venezuela",
'Viet Nam' : "Vietnam"}
def rename_countries(x, country_dict):
    new_name = country_dict.get(x)
    if new_name is not None:
        #print(x, "-->", new_name)
        return new_name
    else:
        return x
wpop = wpop[wpop['Time']==2020].reset_index(drop=True)
wpop['Location'] = wpop.Location.apply(lambda x : rename_countries(x, c
clean_wpop = wpop[wpop['Location'].isin(df['Country_Region'].unique())]
population_distribution = []
for country, gpdf in clean_wpop.groupby("Location"):
    aux = {f"age_{age_grp}": tot for age_grp, tot in zip(gpdf.AgeGrp, g
    aux["Country_Region"] = country
    population_distribution.append(aux)
df_pop_distrib = pd.DataFrame(population_distribution)
# add missing countries with median values
no_data = []
for country in df['Country_Region'].unique():
    if country not in df_pop_distrib['Country_Region'].unique():
```

```
        aux = df_pop_distrib.drop('Country_Region', axis=1).median(axis
        aux["Country_Region"] = country
        no_data.append(aux)
df_no_data = pd.DataFrame(no_data)
df_pop_distrib = pd.concat([df_pop_distrib, df_no_data], axis=0)
# normalize features
norm_pop_distrib = df_pop_distrib.drop("Country_Region", axis=1).div(df
norm_pop_distrib['total_pop'] = df_pop_distrib.drop("Country_Region", a
norm_pop_distrib["Country_Region"] = df_pop_distrib["Country_Region"]
del df_pop_distrib
del df_no_data
del clean_wpop
del wpop
df = df.merge(norm_pop_distrib, on="Country_Region", how='left')
```

In [210]:

```
# Data From: https://ourworldindata.org/smoking#prevalence-of-smoking-a
smokers = pd.read_csv('share-of-adults-who-smoke.csv')
smokers = smokers[smokers.Year == 2016].reset_index(drop=True)
smokers_country_dict = {'North America' : "US",
 'Gambia' : "The Gambia",
 'Bahamas': "The Bahamas",
 "'South Korea'" : "Korea, South",
'Papua New Guinea' : "Guinea",
 "'Czech Republic'" : "Czechia",
 'Congo' : "Congo (Brazzaville)"}
smokers['Entity'] = smokers.Entity.apply(lambda x : rename_countries(x,
no_datas_smoker = []
for country in df['Country_Region'].unique():
    if country not in smokers.Entity.unique():
        mean_score = smokers[['Smoking prevalence, total (ages 15+) (%
        mean_score['Entity'] = country
        no_datas_smoker.append(mean_score)
no_data_smoker_df = pd.DataFrame(no_datas_smoker)
clean_smoke_data = pd.concat([smokers, no_data_smoker_df], axis=0)[['Er
clean_smoke_data.rename(columns={"Entity": "Country_Region",
                                 "Smoking prevalence, total (ages 15+)
df = df.merge(clean_smoke_data, on="Country_Region", how='left')
```

```
/home/lechen/anaconda3/envs/lechen/lib/python3.7/site-packages/ipykernel_launcher.py:22: FutureW
catenation axis is not aligned. A future version
of pandas will change to not sort by default.

To accept the future behavior, pass 'sort=False'.

To retain the current behavior and silence the warning, pass 'sort=True'.
```

In [211]:

```
# Add Smokers Percentages By Country
# Data From: https://ourworldindata.org/smoking#prevalence-of-smoking-a
smokers = pd.read_csv('share-of-adults-who-smoke.csv')
smokers = smokers[smokers.Year == 2016].reset_index(drop=True)
smokers_country_dict = {'North America' : "US",
 'Gambia' : "The Gambia",
 'Bahamas': "The Bahamas",
 "'South Korea'" : "Korea, South",
'Papua New Guinea' : "Guinea",
 "'Czech Republic'" : "Czechia",
 'Congo' : "Congo (Brazzaville)"}
smokers['Entity'] = smokers.Entity.apply(lambda x : rename_countries(x,
no_datas_smoker = []
for country in df['Country_Region'].unique():
    if country not in smokers.Entity.unique():
        mean_score = smokers[['Smoking prevalence, total (ages 15+) (%
        mean_score['Entity'] = country
        no_datas_smoker.append(mean_score)
no_data_smoker_df = pd.DataFrame(no_datas_smoker)
clean_smoke_data = pd.concat([smokers, no_data_smoker_df], axis=0)[['Er
clean_smoke_data.rename(columns={"Entity": "Country_Region",
                                 "Smoking prevalence, total (ages 15+)
df = df.merge(clean_smoke_data, on="Country_Region", how='left')
```

```
/home/lechen/anaconda3/envs/lechen/lib/python3.7/site-packages/ipykernel_launcher.py:23: FutureW
catenation axis is not aligned. A future version
of pandas will change to not sort by default.

To accept the future behavior, pass 'sort=False'.

To retain the current behavior and silence the warning, pass 'sort=True'.
```

In [212]:

```
# Concatenate Country and Region Province
def concat_country_province(country, province):
    if not isinstance(province, str):
        return country
    else:
        return country+"_"+province

# Concatenate region and province for training
df["Country_Region"] = df[["Country_Region", "Province_State"]].apply(l
```

In [213]:

```
# Add Time Data information from Quarantine, Restrictions and Schools
# Data From: https://www.kaggle.com/koryto/countryinfo
country_info = pd.read_csv('covid19countryinfo.csv')
country_info = country_info[~country_info.country.isnull()].reset_index
country_info.drop([ c for c in country_info.columns if c.startswith("Ur
country_info.drop(columns=['pop', 'sex0', 'sex14', 'sex25', 'sex54', 's
                  axis=1,
                  inplace=True)
# Columns with dates
country_info["quarantine"] = pd.to_datetime(country_info["quarantine"])
country_info["restrictions"] = pd.to_datetime(country_info["restrictior
country_info["schools"] = pd.to_datetime(country_info["schools"])
same_state = []
for country in df["Province_State"].unique():
    if country in country_info.country.unique():
        same_state.append(country)
    else:
        pass
country_to_state_country = {}
for state in same_state:
    #print(state)
    #print(df[df["Province/State"]==state]["Country/Region"].unique())
    #print("----")
    country_to_state_country[state] = df[df["Province_State"]==state]["
country_info['country'] = country_info.country.apply(lambda x : rename_
coutry_merge_info = country_info[["country", "density", "urbanpop", "ho
cols_median = ["density", "urbanpop", "hospibed", "lung", "femalelung",
coutry_merge_info.loc[:, cols_median] = coutry_merge_info.loc[:, cols_m
merged = df.merge(coutry_merge_info, left_on="Country_Region", right_or
merged.loc[:, cols_median] = merged.loc[:, cols_median].apply(lambda x:
country_dates_info = country_info[["country", "restrictions", "quaranti
def update_dates(a_df, col_update):
    """
    This creates a boolean time series with one after the start of conf
    """
    gpdf = a_df.groupby("Country_Region")
    new_col = gpdf.apply(lambda df : df[col_update].notnull().cumsum())
    a_df[col_update] = new_col
for col in ["restrictions", "quarantine", "schools"]:
    print(merged.shape)
```

```
    merged = merged.merge(country_dates_info[["country", col]],
                          left_on=["Country_Region", "Date"],
                          right_on=["country", col],
                          how="left",
                          )
    update_dates(merged, col)
drop_country_cols = [x for x in merged.columns if x.startswith("country
merged.drop(columns=drop_country_cols, axis=1, inplace=True)
```

```
(3354, 37)
(3354, 39)
(3354, 41)
```

In [214]:

```
merged.to_csv('enriched_covid_19.csv', index=None)
merged
```

	Unnamed: 0	Province_State	Country_Region	Date	ConfirmedCases	Fatalities	age_0-4
0	0	NaN	Afghanistan	2020-03-19	22	0	0.145717
1	1	NaN	Afghanistan	2020-03-20	24	0	0.145717
2	2	NaN	Afghanistan	2020-03-21	24	0	0.145717
3	3	NaN	Afghanistan	2020-03-22	40	1	0.145717
4	4	NaN	Afghanistan	2020-03-23	40	1	0.145717
...	...	...	...	...	...	...	...
3349	3349	NaN	Malawi	2020-03-27	0	0	0.152846
3350	3350	NaN	Malawi	2020-03-28	0	0	0.152846
3351	3351	NaN	Malawi	2020-03-29	0	0	0.152846
3352	3352	NaN	Malawi	2020-03-30	0	0	0.152846
3353	3353	NaN	Malawi	2020-03-31	0	0	0.152846

3354 rows × 39 columns

In [ ]:
```
1
```