Open in Colab
(https://colab.research.google.com/github/Elzawawy/covid-case-estimator/blob/master/Global_Weather_Data_Exploration.ipynb)

Global Weather Data Exploration

In this dataset we set to explore weather features for our daily cases model estimator.

One of the most important daily features that could have high influence on COVID-19 cases is the weather and its related features like temperature, humidity, wind,.etc. So, and under the light of the previous work conclusions we wish to analyze the weather and temperature data of the respective countries since the outbreak of the virus.|

In [1]:

```
#imports cell
import pandas as pd
import numpy as np
import pickle
from shutil import copyfile

# mount google drive to copy files from repo into drive.
from google.colab import drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6
bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redirect_uri=urn%3aietf%3awg%3aoauth
%3a2.0%3aoob&response_type=code&scope=email%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdocs.te
st%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive%20https%3a%2f%2fwww.googleapis.com%2fauth%
2fdrive.photos.readonly%20https%3a%2f%2fwww.googleapis.com%2fauth%2fpeopleapi.readonly

Enter your authorization code:
··········
Mounted at /content/drive
```

Downloading Dataset

- We use the Official API for https://www.kaggle.com (https://www.kaggle.com) to get our datasets.
- You can get your own Kaggle API key to run this cell by going to kaggle.com and navigating to `My Account` Tab and use the `Create API Key` button, you then upload it to the notebook's temproray storage.
- The first dataset we explore is the Global Weather Dataset for COVID-19 by Pierre Winter (https://www.kaggle.com/winterpierre91/covid19-global-weather-data)

In [3]:

```
!pip install kaggle
# You have to upload you own Kaggle API which is the `kaggle.json` into the temp directory first.
!cp /content/kaggle.json ~/.kaggle/kaggle.json
# For the Kaggle API key to be un-readable by other users on this system.
!chmod 600 /root/.kaggle/kaggle.json
!kaggle datasets download -d winterpierre91/covid19-global-weather-data
!unzip covid19-global-weather-data.zip
!rm covid19-global-weather-data.zip
```

```
Requirement already satisfied: kaggle in /usr/local/lib/python3.6/dist-packages (1.5.6)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.6/dist-packages (from k
aggle) (4.0.0)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.6/dist-packages (from
kaggle) (2.8.1)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.6/dist-packages (from kaggle
) (1.12.0)
Requirement already satisfied: requests in /usr/local/lib/python3.6/dist-packages (from kaggle)
(2.23.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.6/dist-packages (from kaggle) (4.
41.1)
Requirement already satisfied: urllib3<1.25,>=1.21.1 in /usr/local/lib/python3.6/dist-packages
(from kaggle) (1.24.3)
Requirement already satisfied: certifi in /usr/local/lib/python3.6/dist-packages (from kaggle)
(2020.4.5.1)
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.6/dist-packages (f
rom python-slugify->kaggle) (1.3)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.6/dist-packages (fro
m requests->kaggle) (3.0.4)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.6/dist-packages (from req
uests->kaggle) (2.9)
Downloading covid19-global-weather-data.zip to /content
  0% 0.00/204k [00:00<?, ?B/s]
100% 204k/204k [00:00<00:00, 62.8MB/s]
Archive:  covid19-global-weather-data.zip
  inflating: temperature_dataframe.csv
```

Reading and Understanding Dataset

- For each country we have useless columns and ones those we actually need.
- There are some countries with mutiple Provinces and thus multiple data points for each day and ones with single data row for each day (required).
- We map those multiple provinces countries into single ones by taking the mean of features of interest across all provinces for each day.
- We also drop these useless columns to us early on before processing the dataframe to save some extra time.

Weather Features

We hope to find some colleration between certain weather metrics and the speed of the number of infections/deaths.

1. **Temperature ('tempC'):** Temperature measured daily in degrees Celsius.
2. **Humidity ('humidity'):** Humidity measured daily, which is the amount of water vapour in the atmosphere or in a gas.
3. **Hours of SunLight ('sunHour'):** Although the daytime length at the Equator remains 12 hours in all seasons, the duration at all other latitudes varies with the seasons. During the winter, daytime lasts shorter than 12 hours; during the summer, it lasts longer than 12 hours.
4. **Wind Speed ('windspeedKmph'):** It is a fundamental atmospheric quantity caused by air moving from high to low pressure, usually due to changes in temperature.

In [4]:

```
GLOBAL_WHEATHER_DATA_FILE = "/content/temperature_dataframe.csv"
STORAGE_DIR = "/content/drive/My Drive/COVID-19/weather-features/"
copyfile(GLOBAL_WHEATHER_DATA_FILE, STORAGE_DIR+"global_weather_data.csv");

temperature_dataframe = pd.read_csv(STORAGE_DIR+"global_weather_data.csv")
temperature_dataframe.head()
```

Out[4]:

	Unnamed: 0	id	province	country	lat	long	date	cases	fatalities	capital	humidity	sunHour	tempC	win
0	0	1	NaN	Afghanistan	33.0	65.0	2020-01-22	0.0	0.0	Kabul	65.0	8.7	-1.0	
1	1	2	NaN	Afghanistan	33.0	65.0	2020-01-23	0.0	0.0	Kabul	59.0	8.7	-3.0	
2	2	3	NaN	Afghanistan	33.0	65.0	2020-01-24	0.0	0.0	Kabul	71.0	7.1	0.0	
3	3	4	NaN	Afghanistan	33.0	65.0	2020-01-25	0.0	0.0	Kabul	79.0	8.7	0.0	
4	4	5	NaN	Afghanistan	33.0	65.0	2020-01-26	0.0	0.0	Kabul	64.0	8.7	-1.0	

Cleaning the dataset and Dictionary Construction

1. Get country names with multiple provinces.
2. Get country names with single provinces.
3. Remove useless columns.
4. Get the Dates Available Range (We know that its from 1-22 till 3-21 but need it represented in code not hard coded)

In [0]:

```
def extract_from_dataset(dataframe):
  # step 1: countires with NaN in province column is dropped and the rest are ones with many provinces.
  countries_with_mutiple_provinces = dataframe.dropna(subset=["province"]).country.unique()
  # get the difference between the 2 dataframes: all countires dataframe and the countires dataframe with mu
tliple provinces we already built dict for above.
  countries_with_single_province = dataframe.merge(dataframe.dropna(subset=["province"]),indicator = True, h
ow='left').loc[lambda x : x['_merge']!='both'].country.unique()
  # step 3: remove un-needed columns from dataframe in place.
  dataframe = dataframe.drop(columns=["Unnamed: 0","id","lat","long","cases","fatalities","capital","provinc
e"])
  # step 4: get the avaiable date range (22-1 to 21-3) instead of hard-coding it.
  dates_range = dataframe.date.unique()
  return (dataframe,countries_with_mutiple_provinces,countries_with_single_province,dates_range)
```

Create Feature Dictionary Method

1. Calls the `extract_from_dataset` method to prepare dataframe and extract needed smaller Pandas Dataframes Objects.
2. Creates `K:Country- V:Feature` Dictionary for the feature asked for in the params for those countries with multiple provinces first as they need special handling and needs to calculate the mean for their provinces first.
3. Creates `K:Country- V:Feature` Dictionary for the feature asked for in the params for these rest of counties with only one single province which is easier to handle.

Notes about data that had to be handled:

- There were found some countires with no desired features, that's why we add the count != 0 check at the second loop.
- There were found one country (Gambia) with duplicated data for each day, that's why we add the drop_duplicates() at the second loop as well.

In [0]:

```
def create_feature_dict(dataframe, feature):
  if(feature not in ['tempC', 'humidity','sunHour', 'windspeedKmph']):
    raise Exception("Feature must be one of the four temperature-related features")
  (dataframe,multi_countries,single_countries,avail_dates) = extract_from_dataset(dataframe)
  # iterate on each country and create a dictionary for feature where the key is the country and the value.
  country_dict = {}
  country_feature = []
  for country in multi_countries:
    # iterate on each date available for this country provinces and get a mean value for them.
    for date in avail_dates:
      country_feature.append(dataframe[(dataframe['country'] == country) & (dataframe['date'] == date)].mean
()[feature])
    country_dict[country] = np.array(country_feature)
    country_feature.clear()

  # iterate on each country and create a dictionary for feature where the key is the country and the value.
  for country in single_countries:
      # Gambia Data has an issue because all of its dates are repeated two times, so we have to drop duplica
tes.
      feature_series = dataframe[dataframe['country'] == country].drop_duplicates()[feature]
      # escape counties with no feature data.
      if(feature_series.count() != 0):
        country_dict[country] = feature_series.to_numpy()
  return country_dict

def save_dict_to_pickle(dict, pickle_file):
  with open(pickle_file, 'wb') as handle:
    pickle.dump(dict, handle, protocol=pickle.HIGHEST_PROTOCOL)
```

Finally, we iterate on each feature from our four weather features.

- Obtain a country and feature dictionary, where the key is the string country name and the value is 1-darray of values ranging from Day 1 till Day 60. (i.e 1-D array with 60 values)
- Save that dictionary into a csv file in permanent google drive storage for later use.

In [0]:

```
for feature in ['tempC', 'humidity','sunHour', 'windspeedKmph']:
  country_feature_dict = create_feature_dict(temperature_dataframe, feature)
  save_dict_to_pickle(country_feature_dict, STORAGE_DIR+feature+"_dict.pickle")
```

Country-Level Weather Features

- In the previous cell, we created a daily weather features (i.e feature value for each day for each country)
- This is beneficial for Model Class One, but what about the Model Class Two ? In the next cells we will calculate for each country the average weather features we got for each day above.

In [0]:

```
for feature in ['tempC', 'humidity','sunHour', 'windspeedKmph']:
  country_feature_dict = create_feature_dict(temperature_dataframe, feature)
  for country,feature_vector in country_feature_dict.items():
    country_feature_dict[country] = feature_vector.mean()
  save_dict_to_pickle(country_feature_dict, STORAGE_DIR+"country-features/"+feature+"_dict.pickle")
```