K-Means clustering, covid-19

In [1]:

```
import os, re, glob
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import folium
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from IPython.display import IFrame
from IPython.display import Image
from tqdm import tqdm
import warnings
warnings.filterwarnings('ignore')

%matplotlib inline
```

Winter

In [2]:

```
root = 'Covid'
recent_date = "02-01-2021"
previous_date = "01-01-2021"

duplicate_columns = {"Lat": "Latitude",
                     "Long_": "Longitude",
                     "Incidence_Rate": "Incident_Rate",
                     "Case-Fatality_Ratio": "Case_Fatality_Ratio",
                     "Province/State": "Province_State",
                     "Country/Region": "Country_Region",
                     "Last Update": "Last_Update"}

recent_df = pd.read_csv(os.path.join(root, (recent_date + ".csv")))
previous_df = pd.read_csv(os.path.join(root, (previous_date + ".csv")))

for key, value in duplicate_columns.items():
    if key in recent_df.columns:
        recent_df = recent_df.rename(columns={key: value})
    if key in previous_df.columns:
        previous_df = previous_df.rename(columns={key: value})
```

In [3]:

```
recent_df.head()
```

Out[3]:

	FIPS	Admin2	Province_State	Country_Region	Last_Update	Latitude	Longitude	Confirm
0	NaN	NaN	NaN	Afghanistan	2021-02-02 05:22:49	33.93911	67.709953	550
1	NaN	NaN	NaN	Albania	2021-02-02 05:22:49	41.15330	20.168300	789
2	NaN	NaN	NaN	Algeria	2021-02-02 05:22:49	28.03390	1.659600	1075
3	NaN	NaN	NaN	Andorra	2021-02-02 05:22:49	42.50630	1.521800	99
4	NaN	NaN	NaN	Angola	2021-02-02 05:22:49	-11.20270	17.873900	198

In [4]:

```
previous_df.head()
```

Out[4]:

	FIPS	Admin2	Province_State	Country_Region	Last_Update	Latitude	Longitude	Confirm
0	NaN	NaN	NaN	Afghanistan	2021-01-02 05:22:33	33.93911	67.709953	525
1	NaN	NaN	NaN	Albania	2021-01-02 05:22:33	41.15330	20.168300	583
2	NaN	NaN	NaN	Algeria	2021-01-02 05:22:33	28.03390	1.659600	998
3	NaN	NaN	NaN	Andorra	2021-01-02 05:22:33	42.50630	1.521800	81
4	NaN	NaN	NaN	Angola	2021-01-02 05:22:33	-11.20270	17.873900	175

In [5]:

```
current_df = pd.DataFrame(columns=['Province_State','Country_Region','Confirm
current_df['Province_State'] = recent_df['Province_State']
current_df['Country_Region'] = recent_df['Country_Region']
current_df['Confirmed'] = recent_df['Confirmed'] - previous_df['Confirmed']
current_df['Deaths'] = recent_df['Deaths'] - previous_df['Deaths']
current_df['Confirmed'] = recent_df['Confirmed'].astype(int)
current_df['Deaths'] = recent_df['Deaths'].astype(int)
```

In [6]:

```
current_df.shape
```

Out[6]:

```
(4014, 4)
```

In [7]:

```
current_df.head()
```

Out[7]:

	Province_State	Country_Region	Confirmed	Deaths
0	NaN	Afghanistan	55059	2404
1	NaN	Albania	78992	1393
2	NaN	Algeria	107578	2894
3	NaN	Andorra	9972	101
4	NaN	Angola	19829	466

In [8]:

```
name_number = 'EleanorOjo-Emovon-4685.csv'
current_df.to_csv(name_number, index=False)
```

In [9]:

```
data = pd.read_csv(name_number)
```

In [10]:

```
data.head()
```

Out[10]:

	Province_State	Country_Region	Confirmed	Deaths
0	NaN	Afghanistan	55059	2404
1	NaN	Albania	78992	1393
2	NaN	Algeria	107578	2894
3	NaN	Andorra	9972	101
4	NaN	Angola	19829	466

In [11]:

```
print(data.shape)
```

```
(4014, 4)
```

- There are total 4014 rows available in the dataset.

In [12]:

```
print(data.count())
```

```
Province_State     3835
Country_Region     4014
Confirmed          4014
Deaths             4014
dtype: int64
```

In [13]:
```
data.isnull().sum()
```

Out[13]:
```
Province_State    179
Country_Region      0
Confirmed           0
Deaths              0
dtype: int64
```

Q1.

- There are 179 null values in the Province_State

In [14]:
```
data.describe()
```

Out[14]:

	Confirmed	Deaths
count	4.014000e+03	4014.000000
mean	2.585146e+04	581.337070
std	1.289825e+05	3441.235551
min	0.000000e+00	0.000000
25%	9.042500e+02	13.000000
50%	2.517000e+03	42.000000
75%	9.719000e+03	148.000000
max	3.358064e+06	114158.000000

In [15]:
```
data.loc[data['Province_State'].isnull(),'Province_State'] = data['Country_Re
```

In [16]:
```
data.head()
```

Out[16]:

	Province_State	Country_Region	Confirmed	Deaths
0	Afghanistan	Afghanistan	55059	2404
1	Albania	Albania	78992	1393
2	Algeria	Algeria	107578	2894
3	Andorra	Andorra	9972	101
4	Angola	Angola	19829	466

In [17]:
```
states = data['Province_State'].unique()
print("Number of unique States - ", len(states))
```

```
Number of unique States -  773
```

In [18]:
```
country = data['Country_Region'].unique()
print("Number of unique Countries- ", len(country))
```

```
Number of unique Countries-  201
```

Q2. This is my answer;

- There are 201 unique ciuntries in the dataset.

In [19]:

```
pip install geopy
```

```
Requirement already satisfied: geopy in c:\users\user\anaconda3\lib\site-pa
ckages (2.3.0)
Requirement already satisfied: geographiclib<3,>=1.52 in c:\users\user\anac
onda3\lib\site-packages (from geopy) (2.0)
Note: you may need to restart the kernel to use updated packages.
```

In [20]:

```
import datetime, time, requests
from time import sleep
from geopy.geocoders import Nominatim


def get_lat_lon(place):
    geolocator = Nominatim(user_agent=name_number)
    location = geolocator.geocode(place)
    lat_lon = location.latitude, location.longitude

    output = [float(i) for i in lat_lon]
    return output
```

In [21]:

```
data['Province_State'].value_counts()
```

Out[21]:

```
Texas                  255
Georgia                162
Virginia               134
Kentucky               121
Missouri               117
                      ...
Manipur                  1
Meghalaya                1
Mizoram                  1
Nagaland                 1
Pitcairn Islands         1
Name: Province_State, Length: 773, dtype: int64
```

In [22]:

```
data['Country_Region'].value_counts()
```

Out[22]:

```
US          3277
Russia        83
Japan         49
India         37
Colombia      34
            ...
Haiti          1
Holy See       1
Honduras       1
Hungary        1
Tuvalu         1
Name: Country_Region, Length: 201, dtype: int64
```

In [23]:

```
from tqdm import tqdm

geo_lat = []
geo_lon = []

not_found = []
found = []
for state in tqdm(states):
    time.sleep(0.2)
    lat_lon = [None, None]
    try:
        lat_lon = get_lat_lon(state)
        found.append(state)
    except:
        not_found.append(state)

    geo_lat.append(lat_lon[0])
    geo_lon.append(lat_lon[1])

if len(not_found) > 0:
    print("Locations are not found for - ", not_found)
else:
    print("Found all the locations")

#if len(found) > 0:
    print("Locations are found for - ", found)
```

```
100%|████████████████████████████████████████████████████████████████
██████████| 773/773 [06:31<00:00,  1.97it/s]

Locations are not found for -  ['Repatriated Travellers', 'Sakha (Yakutiya)
Republic', 'Summer Olympics 2020', 'W.P. Kuala Lumpur']
```

In [24]:

```
states_list = states.tolist() #converting states to list to index list's item
lats = []
lons = []
for i, r in data.iterrows():
    state = r['Province_State']
    index_list = states_list.index(state)
    lats.append(geo_lat[index_list])
    lons.append(geo_lon[index_list])


data['Latitude'] = lats
data['Longitude'] = lons
```

In [25]:
```
data.head()
```

Out[25]:

	Province_State	Country_Region	Confirmed	Deaths	Latitude	Longitude
0	Afghanistan	Afghanistan	55059	2404	33.768006	66.238514
1	Albania	Albania	78992	1393	41.000028	19.999962
2	Algeria	Algeria	107578	2894	28.000027	2.999983
3	Andorra	Andorra	9972	101	42.540717	1.573203
4	Angola	Angola	19829	466	-11.877577	17.569124

Q3. This is my answer;

- The Latitude and Longitude given in the dataset and the one gotten from geopy are not totally identical but the differences are very minute, an error or from 0.1-1.0 which in some cases can be considered as negligible.

In [26]:
```
#Selected rows without NaN
data = data[data['Latitude'].notna()]
```

In [27]:
```
data.shape
```

Out[27]:
```
(4010, 6)
```

In [28]:
```
clustering_data = data[["Confirmed", "Deaths"]]
```

In [29]:
```
clustering_data.head()
```

Out[29]:

	Confirmed	Deaths
0	55059	2404
1	78992	1393
2	107578	2894
3	9972	101
4	19829	466

In [30]:
```
data.isnull().sum()
```

Out[30]:
```
Province_State    0
Country_Region    0
Confirmed         0
Deaths            0
Latitude          0
Longitude         0
dtype: int64
```

In [31]:
```
scaler = StandardScaler()
X_scaled = scaler.fit(clustering_data).transform(clustering_data.astype(np.fl
```

In [32]:
```
cluster_range = range( 1, 20 )
cluster_errors = []

for num_clusters in cluster_range:
    clusters = KMeans( num_clusters )
    clusters.fit( X_scaled )
    cluster_errors.append( clusters.inertia_ )

clusters_df = pd.DataFrame( { "num_clusters":cluster_range,
                             "cluster_errors": cluster_errors } )

plt.figure(figsize=(16,6))
plt.plot( clusters_df.num_clusters, clusters_df.cluster_errors, marker = "o"
```

```
------------------------------------------------------------------------
--
AttributeError                          Traceback (most recent call las
t)
Input In [32], in <cell line: 4>()
      4 for num_clusters in cluster_range:
      5     clusters = KMeans( num_clusters )
----> 6     clusters.fit( X_scaled )
      7     cluster_errors.append( clusters.inertia_ )
      9 clusters_df = pd.DataFrame( { "num_clusters":cluster_range,
     10                              "cluster_errors": cluster_errors } )

File ~\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1186, in KM
eans.fit(self, X, y, sample_weight)
   1183     print("Initialization complete")
   1185 # run a k-means once
-> 1186 labels, inertia, centers, n_iter_ = kmeans_single(
   1187     X,
   1188     sample_weight,
```

In [ ]:
```
# Fitting K-Means to the dataset
kmeans = KMeans(n_clusters = 4, init = 'k-means++', random_state = 10)
y_kmeans = kmeans.fit_predict(X_scaled)

#beginning of  the cluster numbering with 1 instead of 0
y_kmeans1=y_kmeans+1

# New list called cluster
cluster = list(y_kmeans1)
# Adding cluster to our data set
clustering_data['cluster'] = cluster
```

In [ ]:
```
clustering_data.head(10)
```

In [ ]:
```
kmeans_mean_cluster = pd.DataFrame(round(clustering_data.groupby('cluster').m
kmeans_mean_cluster
```

Q4; - This is my answer

- Cluster 1 has the least amount of confirmned cases and consequently has the keast amount of death , the death percentage is about 1.9% which is very low, most of the infected induviduals survived.
- Cluster 2 had more cases compared to cluster 1 and had a death rate of 2.1% which is relatively small, which infers a lot of the infected individuals survived, although compared to cluster 1 more people died in this cluster.
- Cluster 3 had more cases than cluster 2 and consequently a death rate of 2.7% which is more than cluster 1&2 meaning more people were infected in these areas and more people died.
- Cluster 4 had the highest number of confirmed cases and highest number of deaths with a death rate of 2.82%.

In [ ]:

```
data['cluster'] = cluster
clusters = data[['Province_State', 'cluster']]
clusters.loc[clusters['cluster'] == 2]
```

In [ ]:

```
data['cluster'] = cluster
clusters = data[['Province_State', 'cluster']]
clusters.loc[clusters['cluster'] == 3]
```

In [ ]:

```
data['cluster'] = cluster
clusters = data[['Province_State', 'cluster']]
clusters.loc[clusters['cluster'] == 1]
```

Type *Markdown* and LaTeX: α^2

In [ ]:

```
data['cluster'] = cluster
clusters = data[['Province_State', 'cluster']]
clusters.loc[clusters['cluster'] == 4]
```

In [ ]:

```
data.head()
```

In [ ]:

```
def get_color(cluster_id):
    if cluster_id == 2:
        return 'yellow'
    if cluster_id == 1:
        return 'green'
    if cluster_id == 3:
        return 'orange'
    if cluster_id == 4:
        return 'darkred'

data["color"] = data["cluster"].apply(lambda x: get_color(x))
```

In [ ]:

```
data.head(10)
```

In [ ]:

```
#create a map
this_map = folium.Map(location =[data["Latitude"].mean(),
                                 data["Longitude"].mean()], zoom_start=5)

def plot_dot(point):
    '''input: series that contains a numeric named latitude and a numeric nam
    this function creates a CircleMarker and adds it to your this_map'''
    folium.CircleMarker(location=[point.Latitude, point.Longitude],
                        radius=2,
                        color=point.color,
                        weight=1).add_to(this_map)




#clustered_full.apply(,axis=1) #use this to iterate through every row in your
data.apply(plot_dot, axis = 1)


#Set the zoom to the maximum possible
this_map.fit_bounds(this_map.get_bounds())

#Save the map to an HTML file
this_map.save(os.path.join('covid_map.html1'))
```

In [ ]:

```
this_map
```

Q5; This is my answer

- From the map France and England have the highest number of confirmed cases of the coronavirus and deaths surrounded by the 2 clusters that have the least number of confirmed cases 'yellow' and 'green' which does not look so right but in 2021 there was a published red list of country that could move in and out for United Kingdom so there was some kind of restriction so it curbed the spread of the virus.Also the areas with 'orange' second highest cases is isolated on the map which means there were probably restrictions.(Public Health Scotland,2021).Public Health Scotland (2021) NewsDetail - Fit for Travel. Available at: https://www.fitfortravel.nhs.uk/news/newsdetail.aspx?id=24150 (https://www.fitfortravel.nhs.uk/news/newsdetail.aspx?id=24150).

K-Means clustering, covid-19

Introduction Covid-19 is an infectious disease caused by Coronavirus that affects the respiratory organs with mild to severe symptoms which could eventually cause death. Viruses like Coronavirus have been proven to be affected by some environmental factors which either stop their propagation or inhibit their growth. According to research, the Coronavirus dies in 5 minutes under 70°C due to low humidity and high temperatures in the air and on surfaces. (Coronavirus dies within five minutes at 70°. (2022). This report is based on analyzing the spread of Covid-19 in the summer and winter of 2021, and although this was more than a year after the outbreak the spread of covid-19 was wild. Summer and winter correspond to hot and cold seasons, so let's see if the virus spreads as much in the summer as in the winter. This is the case with January representing winter and July representing summer.

Data Preparation and Experimentation K-means clustering is used when we do not have a particular outcome variable or target variable but have different features to find similarities and classify them for inference and this is the best machine learning model to see the effect of the Coronavirus on different areas around the world during different times or seasons. The dataset was gotten from GitHub and loaded to Jupiter notebook where the pre-processing processes went on seamlessly. Some of the confirmed cases were represented as floats which were converted to integers, and the null values from the provinces were dealt with before the longitude and latitude were ascribed for the clusters to be formed using geopy which didn’t differ so much from the latitude, and longitude from the dataset, there were 773 provinces and 201 countries. Using the elbow method to get the number of clusters we need for this dataset, we concluded that the number of clusters we need to have is 4 clusters. Analysis was done to classify all provinces into their various clusters. The color scheme for the visualization of the clusters on the map is as thus, dark red for clusters with the highest number of confirmed cases and deaths and green for the clusters with the least number of confirmed cases and deaths, yellow for the second least number and orange for the second highest number. The darker color codes represent more deaths and the lighter color codes represent fewer deaths. The map was displayed using folium and news links were added to backup all the information given by the map.

Results and Conclusions From our winter analysis, cluster 4 had the highest number of confirmed cases and deaths with England and France as the provinces in this cluster. From research, it was seen that January 2021 in the United Kingdom was the coldest since January 2010 (Farrow,2021), simultaneously in Paris 21 cloudy days, 7 days of precipitation, and just 3 sunny days with a very cold degree throughout the month which could be the trigger to the spread of the virus as it spreads like wildfire in the cold where it thrives seamlessly and the death cases. From our summer analysis, cluster 1 had the lowest number of confirmed cases and deaths with 3981 provinces. Taking the first province on the list as a case study, Afghanistan is a very hot country with summer temperatures around 50°C and winter temperatures around 20°C. SCA (2021). From our winter analysis, cluster 1 had the lowest number of confirmed cases and deaths with 3922 provinces, almost the same provinces as cluster 1 in our summer analysis. They are provinces with very hot weather. From our summer analysis, cluster 2 had the highest number of deaths with Indonesia as the only province there which mostly had rain and thunderstorms in July 2021 which is the possible reason for the spread. Santiago, J. (2022)

References Farrow, J. (2021) Looking back at January 2021 and a look ahead to February cold and snow. Available at: https://www.netweather.tv/weather-forecasts/news/10695-looking-back-at-january-2021-and-a-look-ahead-to-february-cold-and-snow (https://www.netweather.tv/weather-forecasts/news/10695-looking-back-at-january-2021-and-a-look-ahead-to-february-cold-and-snow).

Santiago, J. (2022) hikersbay.com. Available at: http://hikersbay.com/climate/july/indonesia?lang=en (http://hikersbay.com/climate/july/indonesia?lang=en).

SCA(2021) Afghanistan’s climate – SCA. Available at: https://swedishcommittee.org/afghanistan/climate/ (https://swedishcommittee.org/afghanistan/climate/).

Chen, S. (2021) Climate and the spread of COVID-19. Available at: https://www.nature.com/articles/s41598-021-87692-z?error=cookies_not_supported&code=9a31cad1-b572-43c6-98c1-574f0cecd8d8

(https://www.nature.com/articles/s41598-021-87692-z?error=cookies_not_supported&code=9a31cad1-b572-43c6-98c1-574f0cecd8d8).

Coronavirus dies within five minutes at 70°C? (2022). Available at: https://www.unicef.org/montenegro/en/stories/coronavirus-dies-within-five-minutes-70c (https://www.unicef.org/montenegro/en/stories/coronavirus-dies-within-five-minutes-70c).

In [ ]:

```
#create a map
this_map = folium.Map(location =[data["Latitude"].mean(),
                                 data["Longitude"].mean()], zoom_start=5)

def plot_dot(point):
    '''input: series that contains a numeric named latitude and a numeric nam
    this function creates a CircleMarker and adds it to your this_map'''
    folium.CircleMarker(location=[point.Latitude, point.Longitude],
                        radius=2,
                        color=point.color,
                        weight=1).add_to(this_map)




#clustered_full.apply(,axis=1) #use this to iterate through every row in your
data.apply(plot_dot, axis = 1)


#Set the zoom to the maximum possible
this_map.fit_bounds(this_map.get_bounds())

#Save the map to an HTML file
this_map.save(os.path.join('covid_map.html1'))
```

In [ ]: