Introduction to Data Science

Coursework 2

Deadline: 12.00, Friday, 1st April 2022 Submission: ELE

Syed Javedhussain

Student ID: 720027410

Copyright and licences

Copyright

Copyright is an automatic right protected in law as soon as a work is produced. The legal owner in the first instance is the creator (author) of the work. Under UK law, Intellectual Property (IP) generated by an employee in the course of their usual work belongs to the employer. However, the University of Exeter waives ownership of copyright in materials of a scholarly nature such as academic journal articles.

When publishing your work, publishers may require you to sign a Copyright Transfer Agreement, this transfers copyright to the publisher. The publisher will then set out what you are allowed to do with that work. For example, many publishers only allow the accepted manuscript to be made available in an institutional repository, not the final published version. Alternatively, you may retain copyright and grant the publisher an exclusive licence to publish and disseminate your work. Author / institutional retention of copyright is a core principle of Plan S.

Copyright licensing

A copyright licence is a form of contract (also known as a 'permissions agreement') based in copyright law. Licences allow authors and creators to retain copyright whilst granting others permissions to use their work. They enable you to modify your copyright terms and allow your work to be copied, distributed, edited, remixed or built upon, within the boundaries of copyright law.

Table of Contents

Part - 1

Basic Stats

1. Count the total number of tweets, describing how you deal with duplicates or other anomalies in the data set.

Getting ready

The Twitter dataset contains the data for the month of June 2021, data is distributed based on the hourly basis. The total dataset is of 720 Json files under zip compression. It's a huge dataset, we need to load the dataset very much carefully else there are chances of Jupyter getting hang.

Good idea is to load only the required fields and once you have read the whole dataset, store it into a csv file so that next time you can use this for future reference.

How to do it...

We need to pull only the required fields from the json files using ***json*** api.

The following are the steps:

1. Iterate though the compression files and at the same time iterate over json files.

```
filenames = next(walk(my_path), (None, None, []))[2]
```

2. Try to read the twitter fields using json api, in the below snippet we are trying to fetch the "tweet" field from the dataset.

```
tweet_list = json.loads(json_obj)["text"]
```

3. Here we are loading the "*text*" field into a list, due to huge dataset if we try to load the data into a Pandas Dataframe, the jupyter will get hang. The best solution is to load the data into a list and then convert back that list into a pandas dataframe.

```
tweet_df = pd.DataFrame(tweet_list,columns=['tweet_data'])
```

4. Now we have a tweets, we can check the length either using the *len()* or the better way we *describe()* the data to get some additional details.

```
print("+++++++++++++++++++++++++++++++++++++++++++++++")
print("Total Number of tweets     : " , total_tweets)
print("Unique Number of tweets    : " , unique_tweets)
print("Duplicate Number of tweets : " , duplicated_tweets)
print("+++++++++++++++++++++++++++++++++++++++++++++++")
```

Output:

```
+++++++++++++++++++++++++++++++++++++++++++++++
Total Number of tweets      :  12018878
Unique Number of tweets     :  11712157
Duplicate Number of tweets  :  306721
+++++++++++++++++++++++++++++++++++++++++++++++
```

Observation

1. So, the total number of tweets are around 12 million, we can see there are duplicate records exists around `306721`, we can remove these duplicates by using the method *drop_duplicates().*

```
twitter_df.drop_duplicates(keep = False, inplace = True)
```

2. For outliers we can use box & whisker plot or scatter plot to analyse what's happening on data and based on that we can drop bad records.
3. The frequency of the tweet repetition is 2588

2. Plot a time-series of the number of tweets by day using the whole dataset and comment on what you see.

Getting ready

In this Twitter dataset we need to plot a Time Series data based on the day (date wise), using the entire dataset and need to observe the pattern like which day most of the people have tweeted.

How to do it...

For this task we need to pull the fields like '*text*' and '*timestamp_ms*' . Here the timestamp_ms field is a long value we need to convert that into a date using the *datetime* api.

The following are the steps:

1. Iterate though the compression files and at the same time iterate over json files, load the dataset into a list and then create a pandas dataframe using that list.

```
      for file in filenames:
             with open(my_path+file) as f:
                    count += 1
                    print("Count = ", count)
                    try:
                      for json_obj in f:
                           text = json.loads(json_obj)['text']
      timestamp =     json.loads(json_obj)['timestamp_ms']
       date_data=str(datetime.datetime.fromtimestamp(float(time
stamp)/1000.0)).split(" ")[0]
                    json_data.append(date_data+"<||>"+text)
                     except:
                         continue
```

Note: Please avoid indentation in this document.

2. Instead of reading the entire dataset set multiple times we can write the data into a csv.

```
      with open("part-1_q2_initial_data.csv", 'w') as f:
             for s in json_data:
                    f.write(str(s) + '\n')
```

3. Create a dataframe, there are chances where we get bad records while reading the file, please pass some additional parameters to resolve those issue as shown below

```
data_df = pd.read_csv('part-1_q2_initial_data_dict.csv',
names=('day', 'tweet'), header=0, sep=',',encoding='utf-8',
error_bad_lines=False)
```

4. Now we have the dataframe with day fields and the tweets, but the day filed column contains the *DD:MM:YYYY HH:MM:SS* format. We need to slice the dates which should satisfy our requirement.

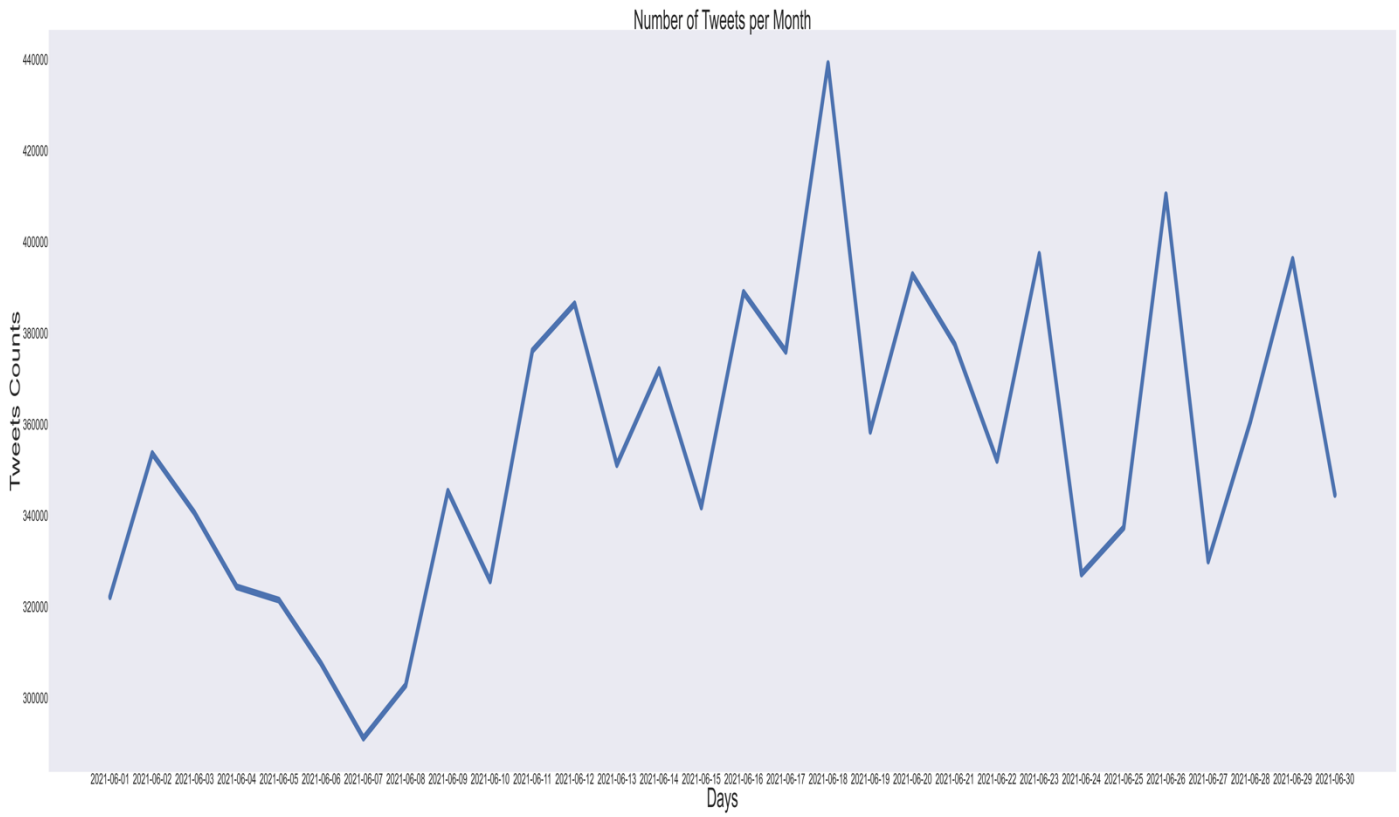
```
      data_df['day_split'] = data_df['day'].str.slice(9, 19)
      data_df['tweet_split'] = data_df['tweet'].str.slice(11,
2)
```

5. Now we need to perform an aggregation of count on tweets to get the results.

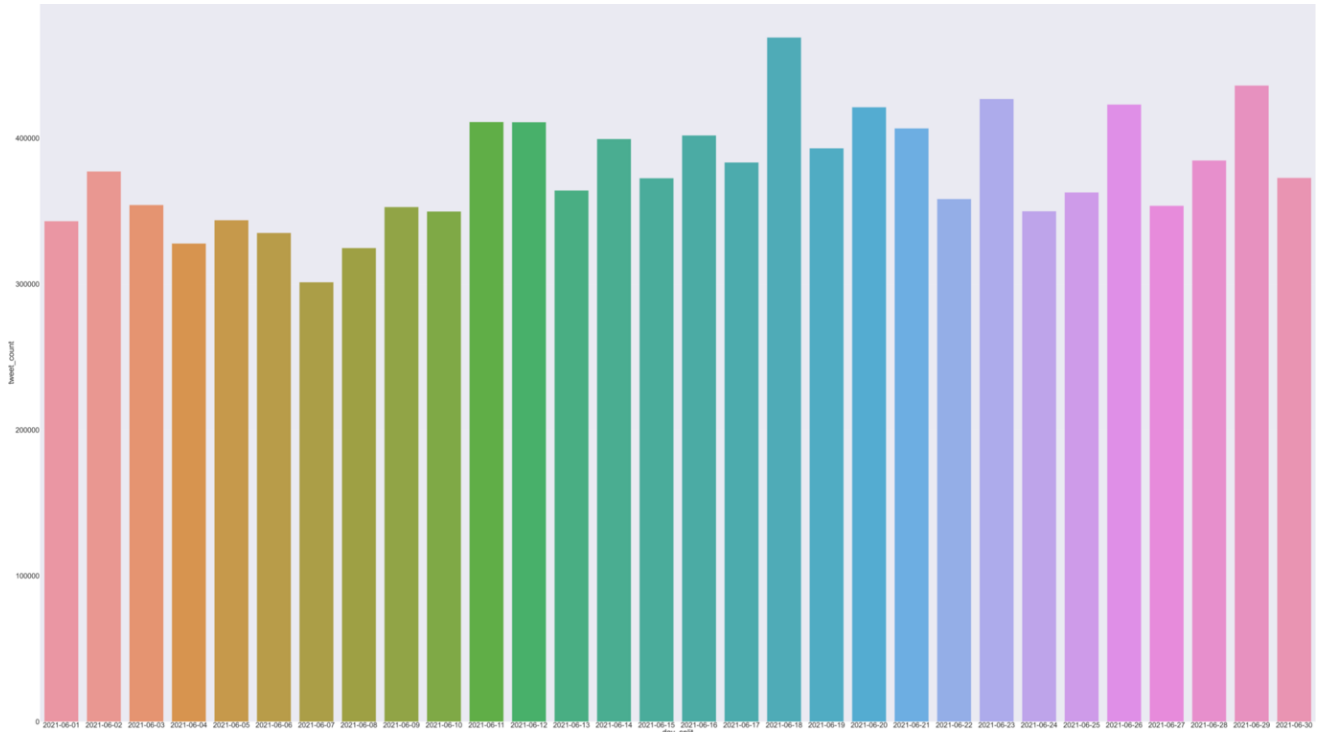
```
data_df_agg =
data_df.groupby('day_split').agg({'tweet_split':['count']})
      data_df_agg.columns = ['tweet_count']
      data_df_agg = data_df_agg.reset_index()
```

6. Once we have the aggregated data, now we can plot timeseries

```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize = (350,100))
plt.rcParams.update({'font.size': 100})
sns.set_style("whitegrid")
sns.set(font_scale = 10)
sns.lineplot(x='day_split', y='tweet_count', data=data_df_agg,
markers=True, dashes=True, linewidth=15, err_style="bars")
plt.xlabel("Days",fontsize=200)
plt.ylabel("Tweets Counts",fontsize=200)
plt.title("Number of Tweets per Month",fontsize=200)
plt.show()
```



7. Bar Plot



8. As per the analysis the high dates where more tweets happen is '2021-06-18', where total number of tweets was 468978. This data is after removing bad records and by dropping the duplicates.

Observation

1. On 2021-06-18 there was a football match between England and Scotland EURO 2020 people were talking about it.
2. People were discussing about *Guterres re-elected for second-term as UN Secretary General*
3. People were discussing about *Serbia's Vucic hits back as Montenegro bans Srebrenica genocide denial*
4. 'Psychic' Thai lion predicts weekend's Euro games
5. Skunk water used against Palestinian protesters

3. Using a box and whisker diagram compare the average number of tweets on the weekdays in the dataset to the numbers for weekend days. Are there statistically significant differences between the number of tweets on weekdays and weekends?

Getting ready

In this Twitter dataset we need to plot a box and whisker diagram based on the average number of tweets on weekdays and weekends, using the entire dataset and need to observe the pattern like in which most of the people have tweeted.

How to do it...

For this task we need to pull the fields like '*text*' and '*timestamp_ms*' . Here the timestamp_ms field is a long value we need to convert that into a date using the *datetime* api.

The following are the steps:

1. Iterate though the compression files and at the same time iterate over json files, load the dataset into a list and then create a pandas dataframe using that list.
2. Instead of reading the entire dataset set multiple times we can write the data into a csv.

3. Create a dataframe, there are chances where we get bad records while reading the file, please pass some additional parameters to resolve those issue as shown below
4. Now we have the dataframe with day fields and the tweets, but the day filed column contains the *DD:MM:YYYY HH:MM:SS* format. We need to slice the dates which should satisfy our requirement.
5. Now we need to split the data in to weekend and weekdays using an pandas function *to_datetime().*

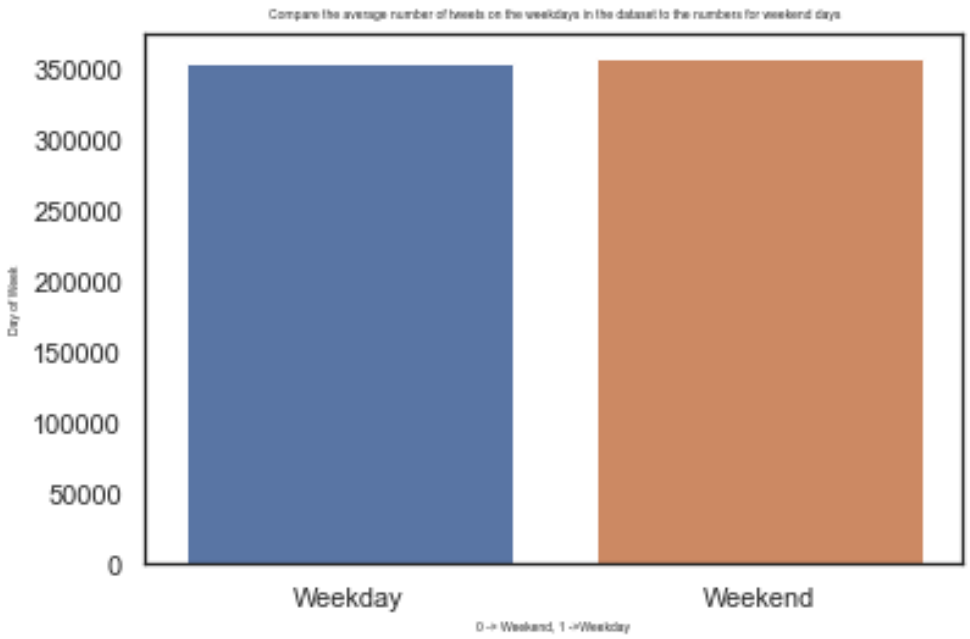
```
data_df_copy['year'] = data_df_copy['day_split'].str.slice(0,4)
data_df_copy['month'] =
data_df_copy['day_split'].str.slice(5,7)
data_df_copy['date'] =
data_df_copy['day_split'].str.slice(8,10)

dates = pd.to_datetime({"year": data_df_copy.year, "month":
data_df_copy.month, "day": data_df_copy.date})

data_df_copy["is_weekend"] = dates.dt.dayofweek > 4

data_df_copy['is_weekend'].replace([True,False],[1,0],inplace=T
rue)
```

6. After performing an aggregation by finding the mean we need to plot the data.

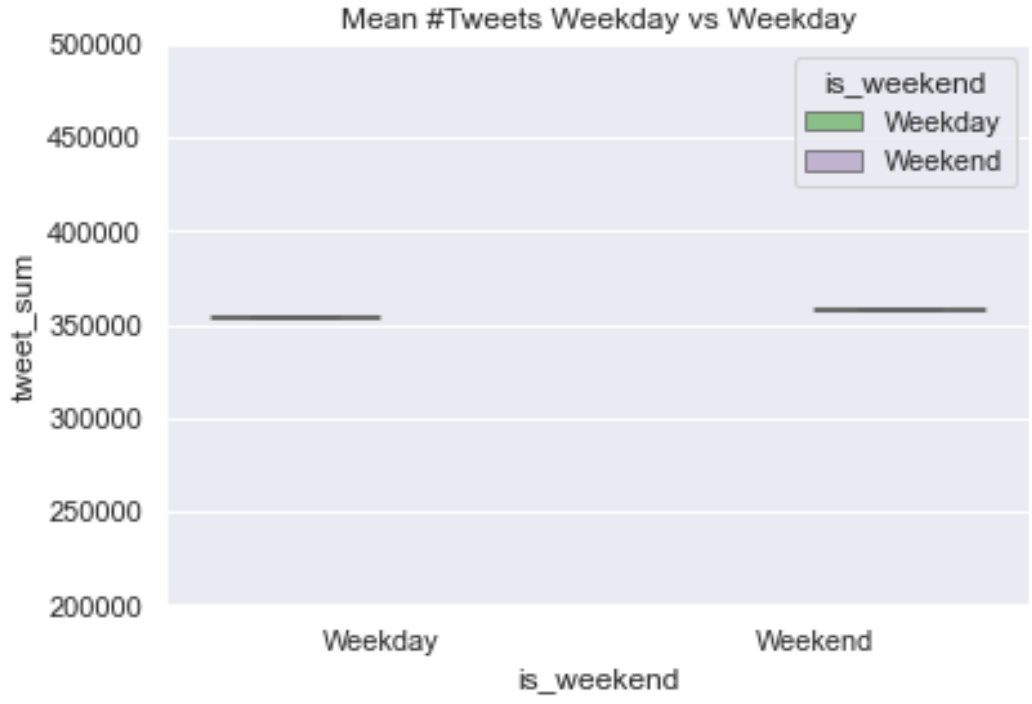
7. Plotting the dataset

```
boxplot = data_df_agg.boxplot(column=['is_weekend',
'tweet_average'])
```

8. Plotting the bar diagram



9. Plotting box plot using s*ns*



Observation

1. As per the box and whisker plot we are getting to know people are tweeting more on weekends when compare to weekdays. I think due to office in weekdays people are busy in their routine stuff and weekends they will get a time to chill out on twitter.

2. If we see the ratio it's almost the figures are nearer but weekends are only 2 days a week and the average is almost more than weekdays.

3. The figures are like

	weekend	tweet_sum
0	Weekday	375710.409091
1	Weekend	380583.375000

4. As we can say in the month of June 2021 weekends (Sat, Sun) people have tweeted more when compare to weekdays (Mon, Tue, Wed, Thu. Fri).

5. Another thought is that the most popular times to tweet could very well correlate to the times when most people are on Twitter. Perhaps it's worth testing also to see if tweeting during a popular time is worthwhile simply for people who are online.

6. In the point-7 graph we can see the median value is almost nearer for both the weekends and weekdays but people are tweeting more on weekends when compare to weekdays.

4. Plot a time-series of the number of tweets by hour, averaged over all weekdays and a similar one for weekend days. Comment on what you see.

Getting ready

In this Twitter dataset we need to plot a tweets using time series plot based on an hour basis. And we need to plot the weekdays and weekends plot by getting the tweets average.

How to do it...

For this task we need to pull the fields like '*text'* and '*timestamp_ms'*. Here the timestamp_ms field is a long value we need to convert that into a date using the *datetime* api. We need to extract the *hour* section from the timestamp.

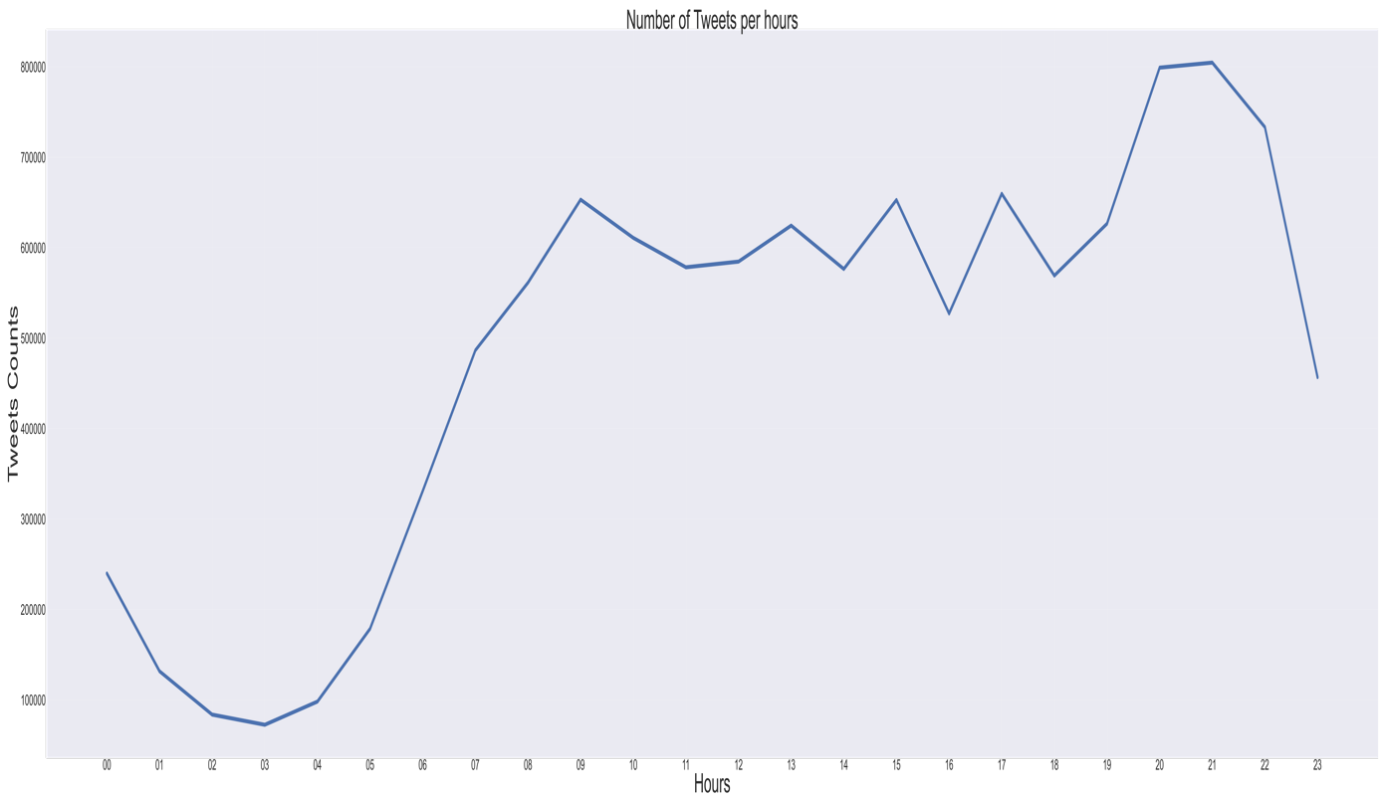
The following are the steps:

1. Iterate though the compression files and at the same time iterate over json files, load the dataset into a list and then create a pandas dataframe using that list.
2. Instead of reading the entire dataset set multiple times we can write the data into a csv.
3. Create a dataframe, there are chances where we get bad records while reading the file, please pass some additional parameters to resolve those issue as shown below
4. Now we have the dataframe with day fields and the tweets, but the day filed column contains the *DD:MM:YYYY HH:MM:SS* format. We need to slice the dates which should satisfy our requirement.
5. Now we need to split the data in to weekend and weekdays using an pandas function *to_datetime().*

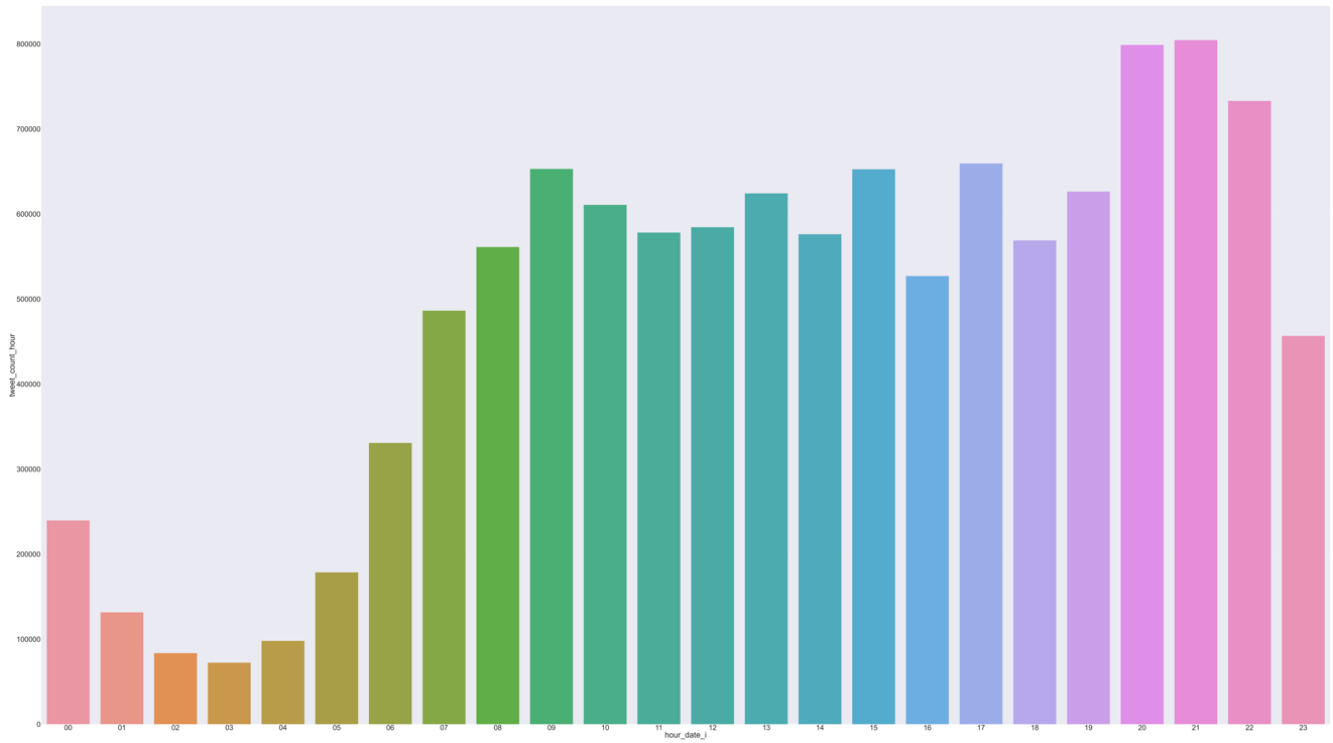
6. For an hour try to get the initial two values from the hours by using the *slice().*

```
data_df_copy['hour_date_i'] =
data_df_copy['hour_date'].str.slice(0, 2)
```

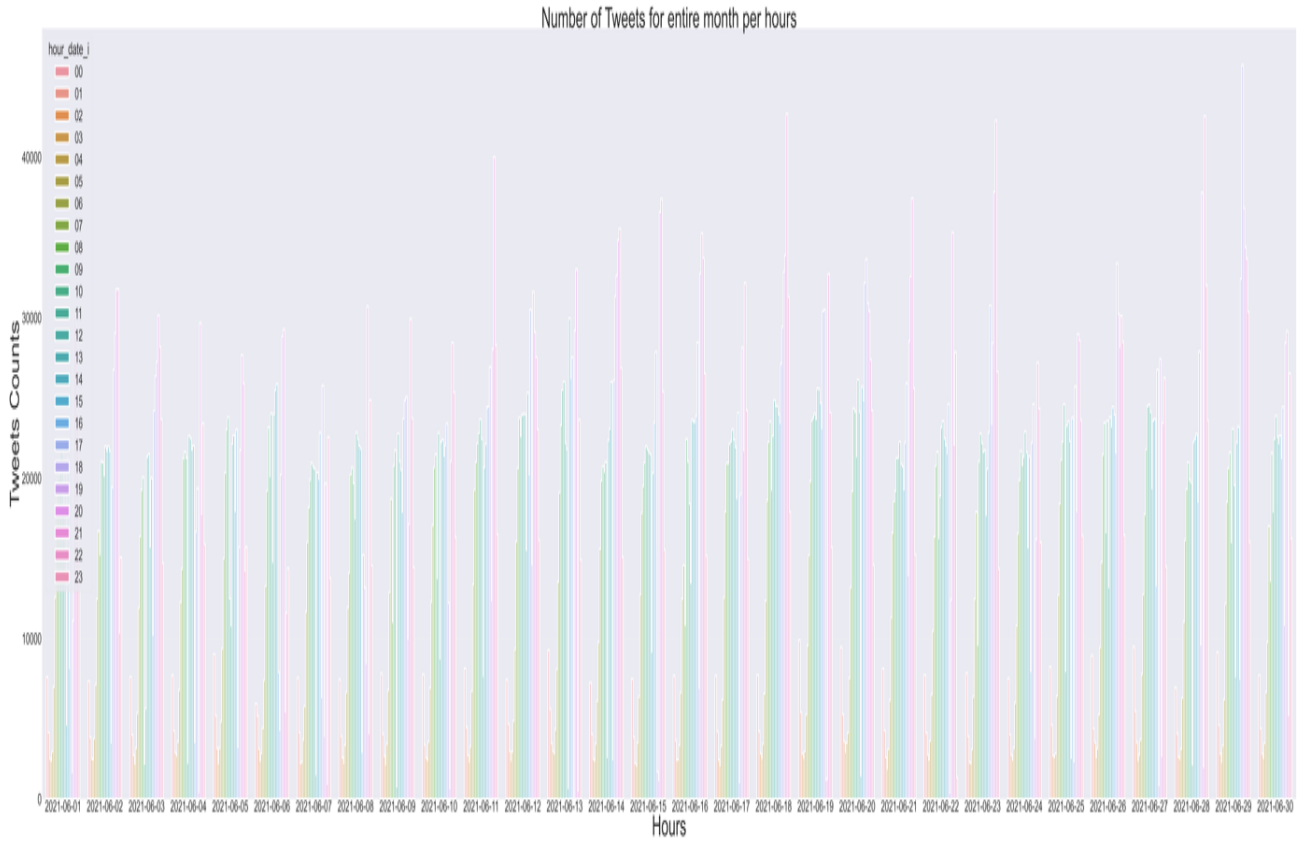
7. Plot the entire month dataset based on the hour basis



8. Plot the hour basis data using a bar diagram

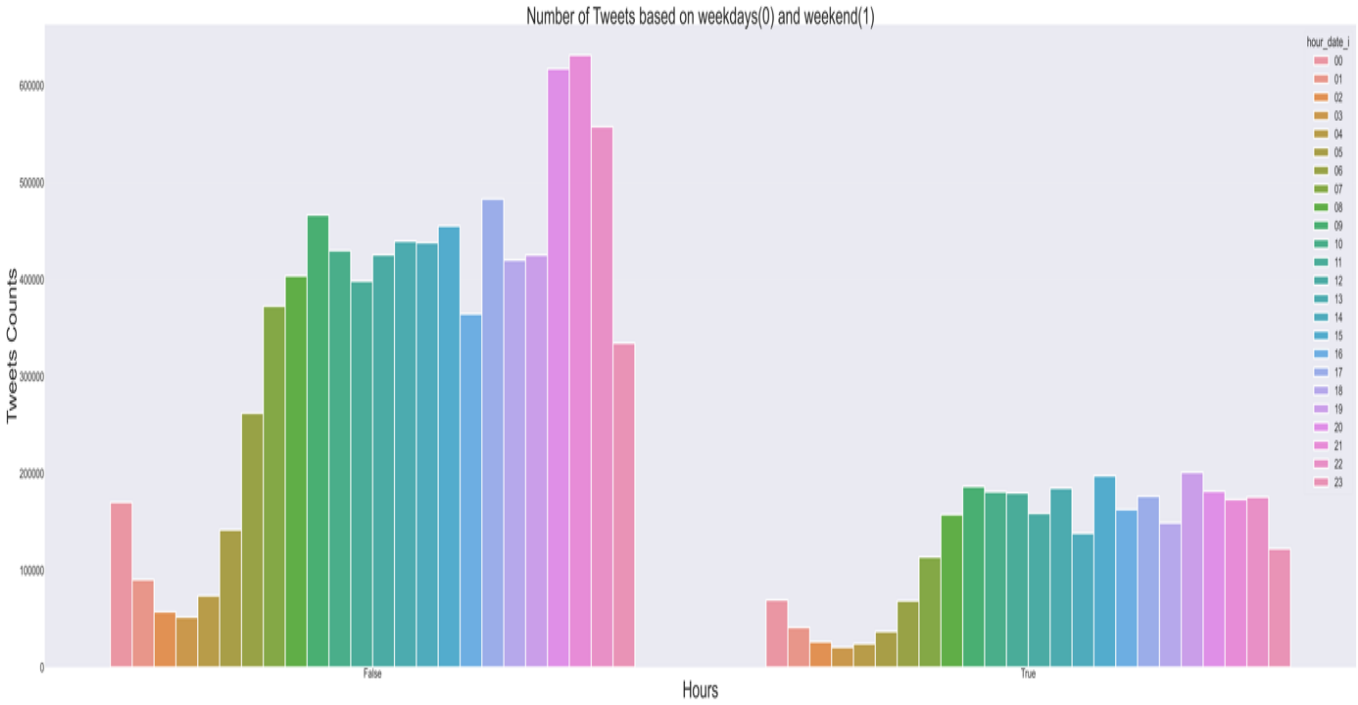


9. Plot a graph on an hourly basis for the whole month



10. Plot the weekdays and weekend data based on the hourly basis.

```
       import matplotlib.pyplot as plt
       import seaborn as sns

       plt.figure(figsize = (350,100))
       plt.rcParams.update({'font.size': 100})
       sns.set_style("whitegrid")
       sns.set(font_scale = 10)
sns.barplot(x='is_weekend', y='tweet_sum', hue='hour_date_i' ,
data=data_df_agg_date_1, linewidth=15)
       plt.xlabel("Hours",fontsize=200)
       plt.ylabel("Tweets Counts",fontsize=200)
       plt.title("Number of Tweets based on weekdays(0) and
weekend(1)",fontsize=200)
       plt.show()
```



Weekday Weekend

Observation

1. In the point-7 time series graph when we are talking about the hourly data based on the entire June month for the year 2021, here we can see the time 8 pm and 9 pm are the peak time where people will sit and do tweeting. It's just a relax time for students, employees etc.

2. In the point-8 bar graph also states the same as above mentioned point, it just a structure of the bar diagram where we can clearly see the plot on an hour basis.

3. In the point-9 graph gives an overall overview of the hourly based data for the entire month, this graph states most of the people tweets during 8 pm ,9 pm and 10 pm.

4. In the point-10 graph shows the hourly average tweets during weekend and weekdays, here we can see during weekdays people are tweeting more on during 8pm and 9pm and on right hand side graph weekend one shows people are 7pm, 8pm.

5. Mostly on the weekend, people plan to go outside and chill out and after coming home they will do a tweet.

Part - 2
Mapping

1. Draw a map of Europe showing the location of the GPS-tagged tweets - these are tweets which have a "coordinates" field in the metadata. The exact form of the map is up to you: marks will be given for accuracy, clarity and presentation.

Getting ready

In this Twitter dataset we need to plot a tweets using *geopandas* and *folium* api. We should be installing these packages and will discuss more about how these api works.

How to do it...

1. Install *geopandas* and *folium* api using the below commands.

```
conda install geopandas or pip install geopandas
conda install -c conda-forge folium or pip install folium
```

2. As this is a geo data we need to fetch the coordinates from twitter dataset, there are two ways you can fetch coordinates either by using *coordinates* tag or a *geo* tag. We need to be careful while fetching the coordinates of the latitude and longitude locations.

3. Once we have a coordinates, load it into a dataframe.

```
data_df = pd.DataFrame(json_data_list,
columns=['latitude','longitude', 'tweet', 'country',
'country_code', 'bounding_box'])
```

4. Typecast latitude and longitude fields to float.

```
data_df['latitude'] =data_df['latitude'].astype(float)
data_df['longitude'] =data_df['longitude'].astype(float)
```

5. Drop the duplicates

```
data_df = data_df.drop_duplicates()
```

6. Now we have a pandas dataframe with the coordinate's points.

7. Now we need to import *geopandas* api and create a geometry points.

```
import pandas as pd
import geopandas as gpd
from shapely.geometry import Point, Polygon
```

8. Create a geometry points field

```
geometry = gpd.points_from_xy(data_df_copy.longitude,
data_df_copy.latitude)
```

9. Create a GeoDataframe containing pandas dataframe and geometry field

```
geo_df =
gpd.GeoDataFrame(data_df_copy[['latitude','longitude','tweet',
'country']], geometry=geometry)

geo_df.head()
```

	latitude	longitude	tweet	country	geometry
0	50.792653	-3.195868	23:45 Temp. 16.3°C, Hum. 84%, Dewp. 12.9°C, Ba...	United Kingdom	POINT (-3.19587 50.79265)
1	53.233430	6.816206	DOING (1:00 uur)	Nederland	POINT (6.81621 53.23343)
2	53.427572	-6.823287	A female cat was lost on 06/06/2021 in Enfield...	Ireland	POINT (-6.82329 53.42757)
3	54.900000	-4.383300	d9da900000p6E9D3	United Kingdom	POINT (-4.38330 54.90000)
4	41.545639	1.893817	És la una en punt de la nit	Espanya	POINT (1.89382 41.54564)

10. As we have a geodataframe with pandas dataframe dataset along with the geometry field which contain POINT. These points will get placed into a map.

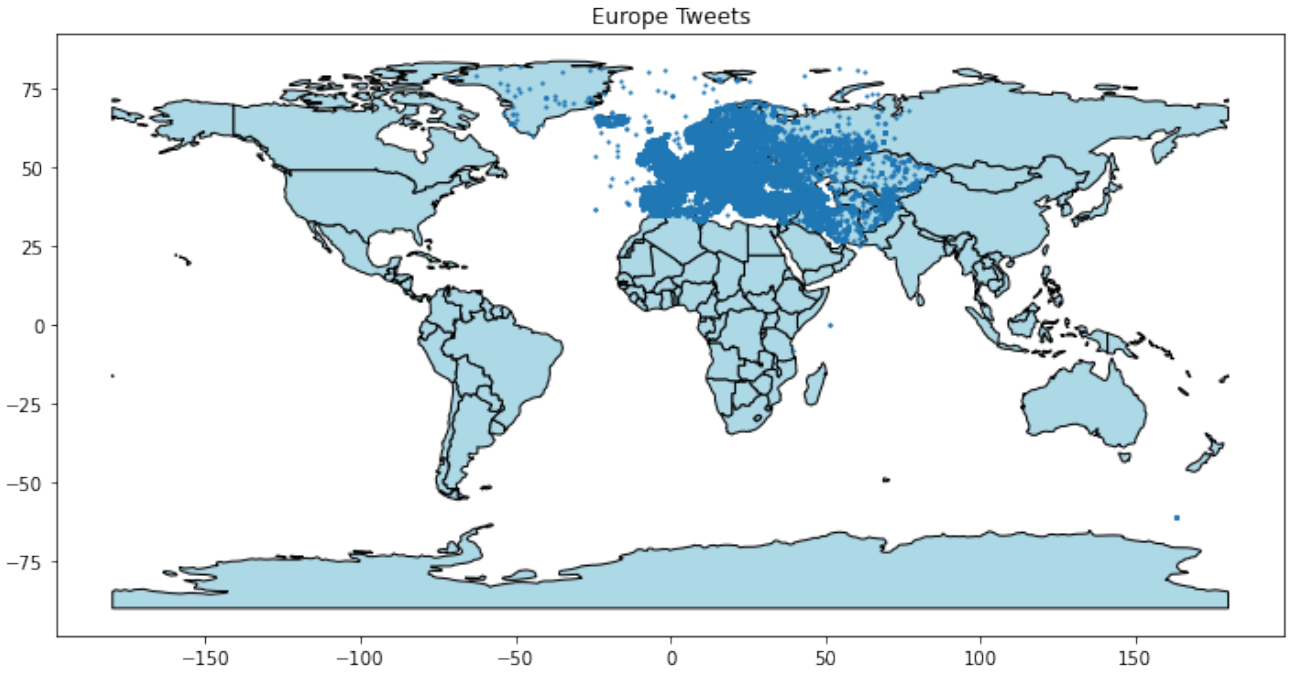
11. As Geodataframe itself provides a world map, let's try first with the world map.

```
# get built in dataset from geopandas
world_data =
gpd.read_file(gpd.datasets.get_path('naturalearth_lowres'))

world_data.head(5)
```

	pop_est	continent	name	iso_a3	gdp_md_est	geometry
0	920938	Oceania	Fiji	FJI	8374.0	MULTIPOLYGON (((180.00000 -16.06713, 180.00000...
1	53950935	Africa	Tanzania	TZA	150600.0	POLYGON ((33.90371 -0.95000, 34.07262 -1.05982...
2	603253	Africa	W. Sahara	ESH	906.5	POLYGON ((-8.66559 27.65643, -8.66512 27.58948...
3	35623680	North America	Canada	CAN	1674000.0	MULTIPOLYGON (((-122.84000 49.00000, -122.9742...
4	326625791	North America	United States of America	USA	18560000.0	MULTIPOLYGON (((-122.84000 49.00000, -120.0000...

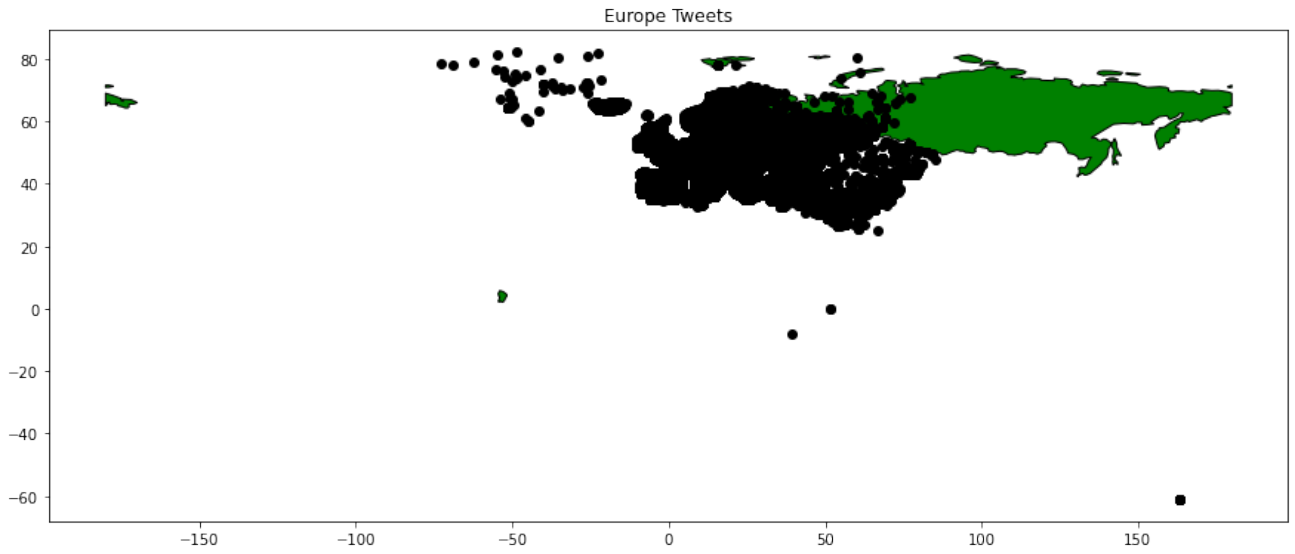
12. Plot the points into the world map

```
axis = world_data.plot(
    color = 'lightblue',
    edgecolor='black'
)


europe_tweets.plot(ax=axis, markersize=1.5, figsize=(10,10))
plt.title("Europe Tweets")

fig = plt.gcf()
fig.set_size_inches(12,8)
fig.savefig('europe_tweet.png', dpi=200)
plt.show()
```



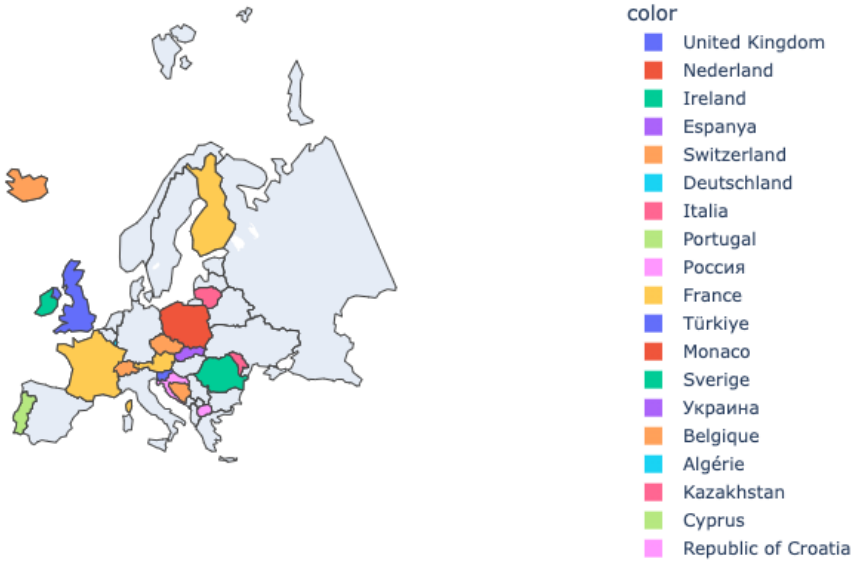
13. We can also apply filter on world map of region wise and plot out points, as shown below

```
axis = world_data[world_data.continent == 'Europe'].plot(
color = 'green', edgecolor='black')

geo_df.plot(ax = axis, color='black')
plt.title('Europe Countries')

fig = plt.cgf()
fig.set_size_inches(9,6)
fig.savefig('country.png', dpi=200)
plt.show()
```



14. If we apply xlim and ylim to have a proper plotting, and also we have grouped by the "*country_code*", so we are taking a base location from all the tweets and plotting into a map

```
f = px.choropleth(locationmode = 'country names',
                  locations=data_df_copy_dedup['country'],
                  scope='europe',
                  color=data_df_copy_dedup['country'])

f.show()
```



15. Now we will try to plot the points in to a Europe Shape files, we can download the shape files from https://ec.europa.eu/eurostat/web/gisco/geodata/reference-data/administrative-units-statistical-units/nuts#nuts21.

16. We can download any CRS (Coordinate Reference System) file but recommended is 4326 format crs file.

17. Once you have downloaded shape files there are other supporting files will be there like project, index, database file, we can just ignore those files.

18. Load the shape file

```
europe_shape_data_nuts =
gpd.read_file('NUTS_LB_2021_4326_Points.shp/NUTS_LB_2021_4326.s
hp')
```

```
europe_tweets.crs
```

Output:

```
 <Geographic 2D CRS: +init=epsg:4326 +type=crs>
Name: WGS 84
Axis Info [ellipsoidal]:
- lon[east]: Longitude (degree)
- lat[north]: Latitude (degree)
Area of Use:
- name: World.
- bounds: (-180.0, -90.0, 180.0, 90.0)
Datum: World Geodetic System 1984 ensemble
- Ellipsoid: WGS 84
- Prime Meridian: Greenwich
```

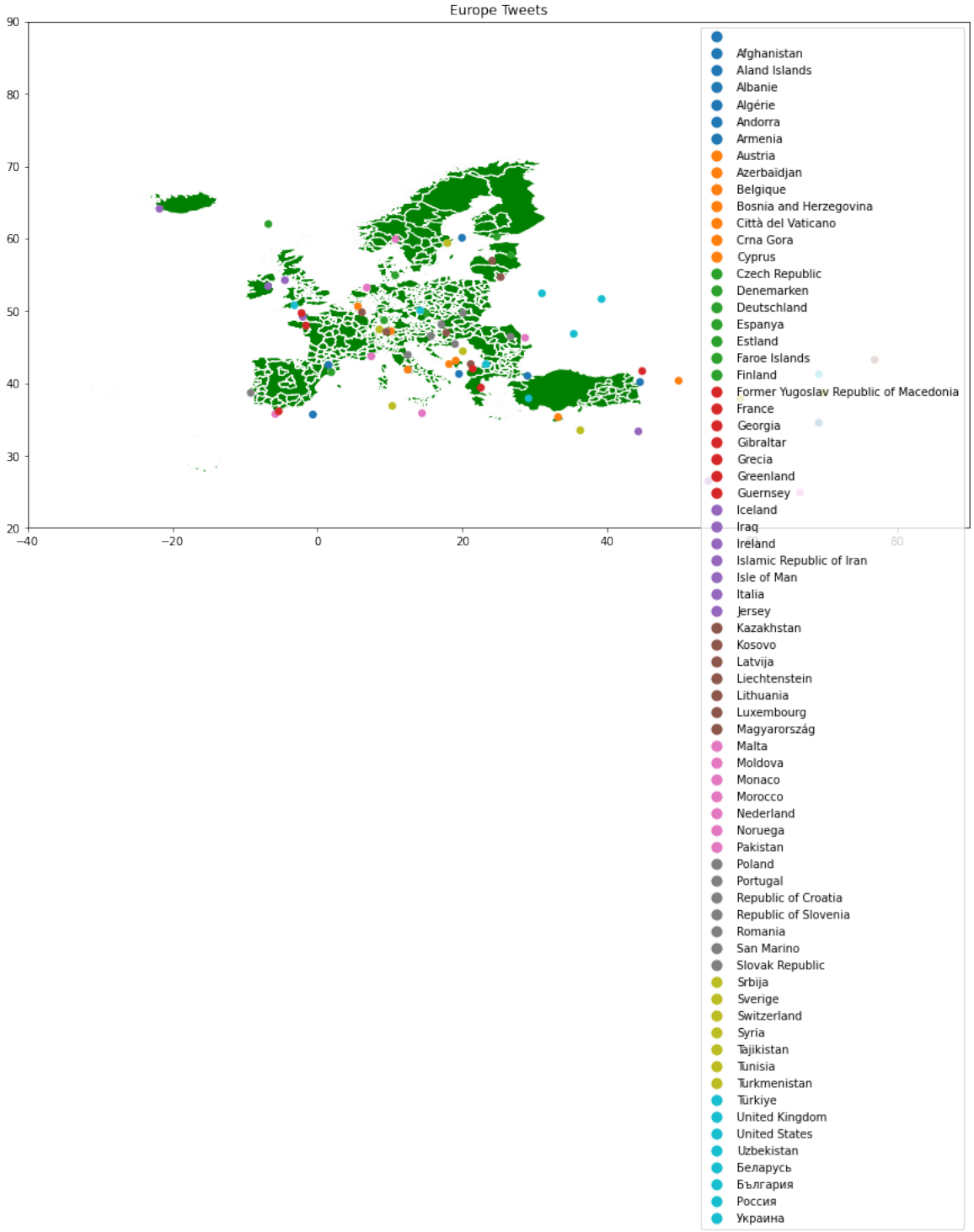
19. Create a geopandas dataframe with the pandas dataframe and geometry field.

```
europe_tweets = gpd.GeoDataFrame(data_df_copy, geometry=points)
europe_tweets.crs = {'init': 'epsg:4326'}
europe_tweets.head()
```

20. Plot the points into a Europe shape file map

```
axis = europe_shape_data_nuts.plot(figsize=(20,20),
    color = 'lightblue',
    edgecolor='green'
```

```
)


#europe_tweets.plot(ax=axis, color='red')
europe_tweets.plot()
#plt.title("Europe Tweets")

#fig = plt.gcf()
#fig.set_size_inches(15,10)
#fig.savefig('europe_tweet.png', dpi=200)
#plt.show()
```



21. Now plot the points using folium, folium will show us the more interactive map and we can also save the map in a vector image.

22. To work with folium with the Europe map we need to download geojson/topojson file.

23. We can download the geojson json file from internet and load it into a folium.

```
geo_str = json.dumps(json.load(open("europe.geojson", 'r'))) #
map data
```

24. Initiate the folium map using the Europe base location as shown below

```
mapeu = folium.Map(location=[48, 11],
                   tiles="CartoDB positron",
                   zoom_start=3)


folium.GeoJson(
    geo_str,
    name='geojson'
).add_to(mapeu)
mapeu
```

25. In the above code we are performing three steps
    a. *Load the Europe geojson file*
    b. *Initiating folium map with the base location Europe.*
    c. *We are providing the polygons from geojson file to the map so that it can add the boundaries.*

26. Now we need to plot our points to the Europe map by iterating our points and adding to the map.

```
for _, r in geo_df.iterrows():
    # Without simplifying the representation of each borough,
    # the map might not be displayed
    sim_geo =
gpd.GeoSeries(r['geometry']).simplify(tolerance=0.001)
    geo_j = sim_geo.to_json()
    geo_j = folium.GeoJson(data=geo_j,
                           style_function=lambda x:
{'fillColor': 'orange'})
    folium.Popup(r['country']).add_to(geo_j)
    geo_j.add_to(mapeu)
mapeu
```

2. Explain any patterns you observe.

1. Based on the GeoPandas and Folium map we can understand that people are tweeting all over the Europe, to get a high glimpse on the map we have performed an aggregation based on the country code and took a base location for that country.
2. The top 5 countries from where we are getting most of the tweets are

country_code	tweet_counts	county_name
GB	143141	United Kingdom
TR	108837	Turkey
ES	79202	Spain
IT	49315	Italy
DE	46202	Deutschland

3. If we plot the top 5 countries based on their tweets

3. The rest of the tweets should have a "place" tag. For these tweets plot the CDF of the bounding box diagonals. Comment on the distribution.

Getting ready

In Twitter dataset we have a *place* tag where we have a bounding box diagonals. We need to find out the CDF (Cumulative distribution function) for those bounding box diagonals.

How to do it...

For this task we need to pull the fields like '*text'* and '*place'*. From the place tag we need to extract the bounding box diagonals.

The following are the steps:

1. Iterate though the compression files and at the same time iterate over json files, load the dataset into a list and then create a pandas dataframe using that list.

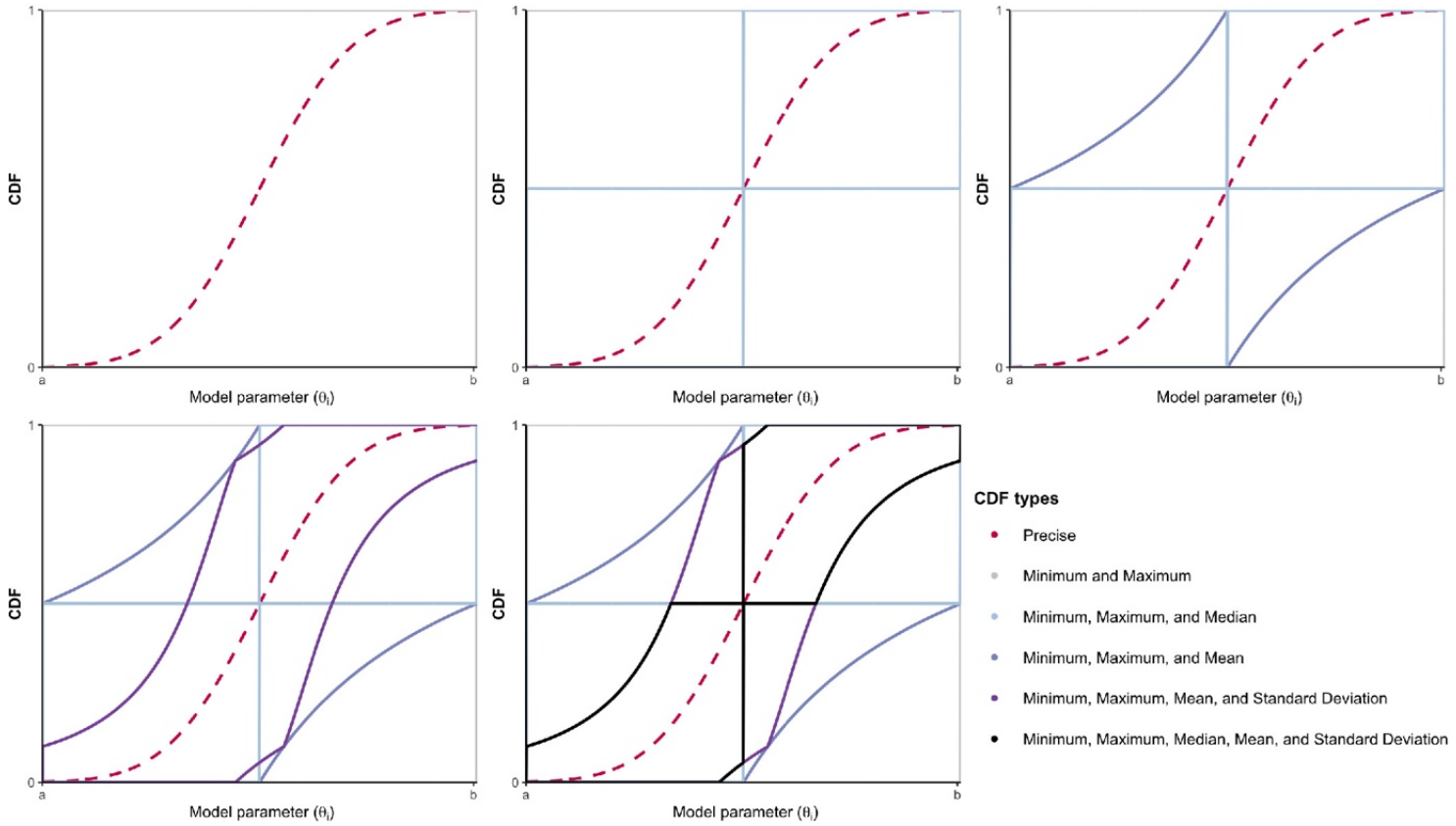
2. Instead of reading the entire dataset set multiple times we can write the data into a csv.

3. Create a dataframe, there are chances where we get bad records while reading the file, please pass some additional parameters to resolve those issue as shown below

4. Pull the bounding box diagonal field from the *place* tag.

```
import matplotlib.pyplot as plt
import numpy as np

# create some randomly ddistributed data:

# sort the data:
data_sorted = np.sort(json_list)
print(data_sorted)
# calculate the proportional values of samples
p = 1. * np.arange(len(json_list)) / (len(json_list) - 1)

# plot the sorted data:
fig = plt.figure()
ax1 = fig.add_subplot(121)
ax1.plot(p, data_sorted)
ax1.set_xlabel('CDF')
```

```
ax1.set_ylabel('Model Parameter')

ax2 = fig.add_subplot(122)
ax2.plot(data_sorted, p)
ax2.set_xlabel('CDF')
ax2.set_ylabel('Model Parameter')
```



Part - 3

Users

1. **Make a histogram of tweets per user with number of users on the y-axis and number of tweets they make on the x-axis. Discuss the distribution that you see. All the users in the data set should be included!**

Getting ready

In Twitter dataset we have a user's profile data where we can track his tweets, followers, friend, location etc. As part of this task, we need to plot a histogram by getting the number of tweets tweeted by number of users.

How to do it...

For this task we need to pull the fields like '*text*', '*name', 'id', 'country', 'place*'.

The following are the steps:

1. Iterate though the compression files and at the same time iterate over json files, load the dataset into a list and then create a pandas dataframe using that list.

2. Instead of reading the entire dataset set multiple times we can write the data into a csv.

3. Create a dataframe, there are chances where we get bad records while reading the file, please pass some additional parameters to resolve those issue as shown below

4. Now we have the dataframe with user name, user screen name, user id, place, country fields and the tweets.

5. Perform a grouping operation based on the user_id and get the count of his tweets

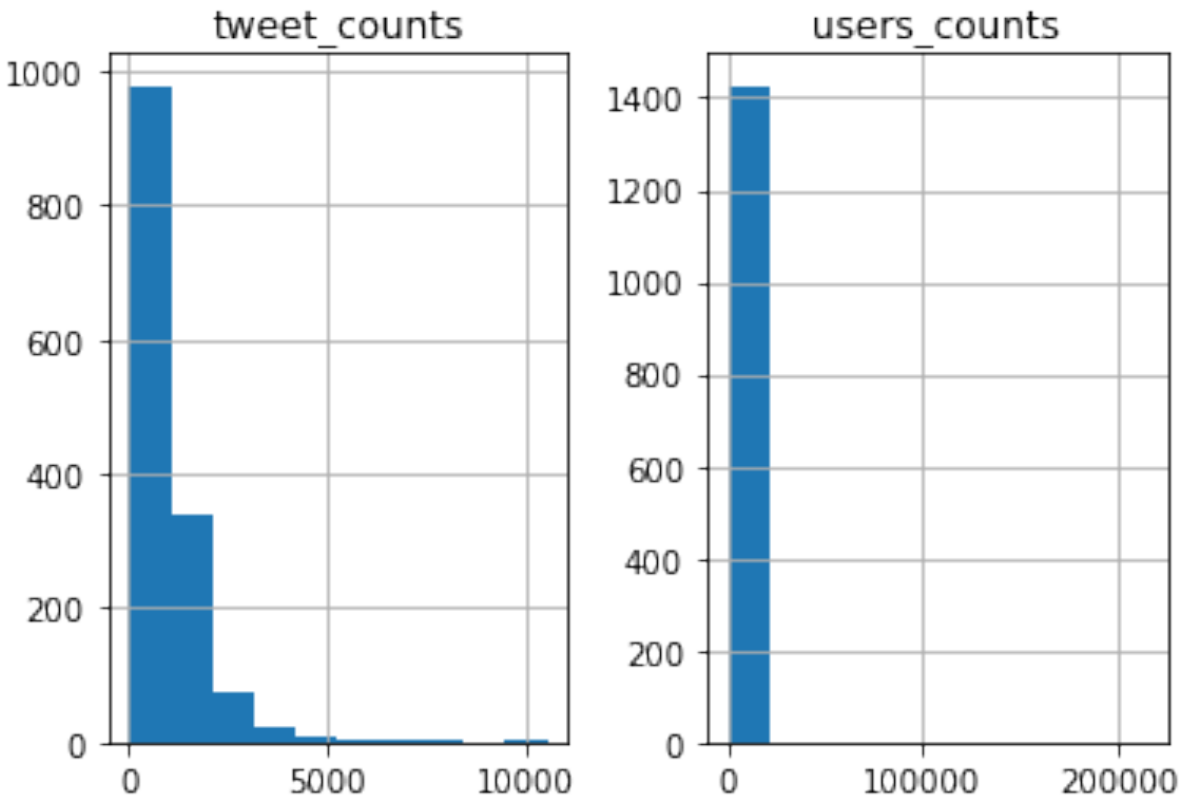
```
data_df_user_id_tweet_count =
data_df.groupby('user_id').agg({'text':['count']})
data_df_user_id_tweet_count.columns = ['tweet_counts']
data_df_user_id_tweet_count =
data_df_user_id_tweet_count.reset_index()
```

6. Next, we need to perform second level of grouping based on the "tweet_counts" and performing an aggregation on the "user_id"

```
data_df_user_id_users_count =
data_df_user_id_tweet_count.groupby('tweet_counts').agg({'user_
id':['count']})
data_df_user_id_users_count.columns = ['users_counts']
data_df_user_id_users_count =
data_df_user_id_users_count.reset_index()
```

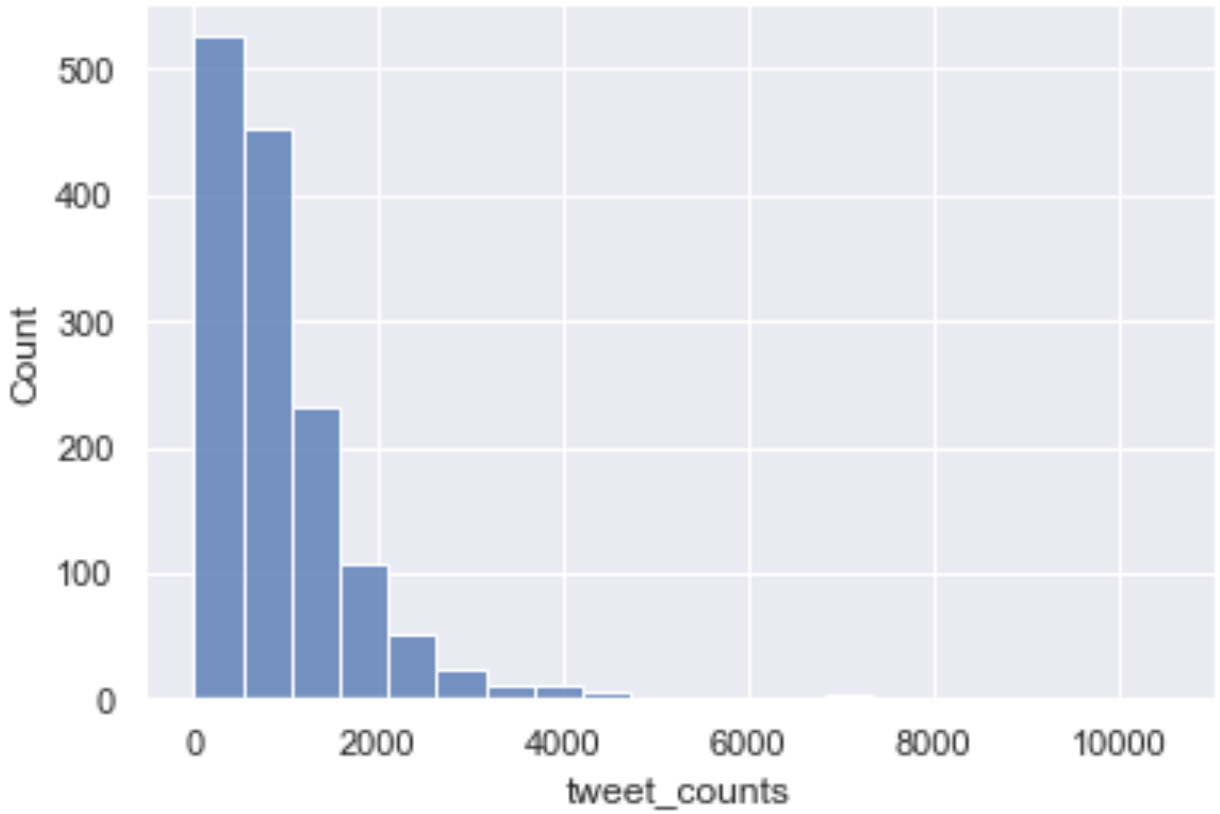
7. Now once we have the aggregated data, let's do a histogram plot.

```
data_df_user_id_users_count.hist()
```



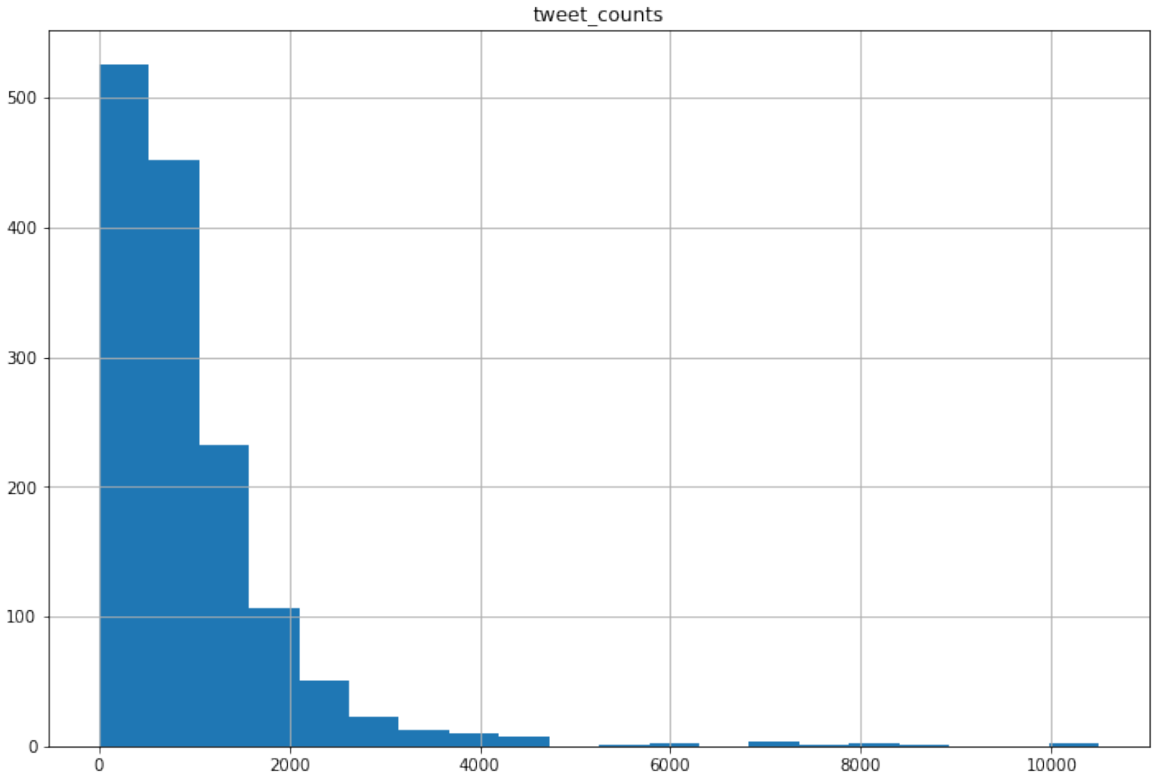
8. Plot the histogram based on the tweets counts

```
import seaborn as sns
import matplotlib.pyplot as plt
# set a grey background (use sns.set_theme() if seaborn version
0.11.0 or above)
sns.set(style="darkgrid")
sns.histplot(data=data_df_user_id_users_count,
x="tweet_counts", bins=20)
plt.show()
```



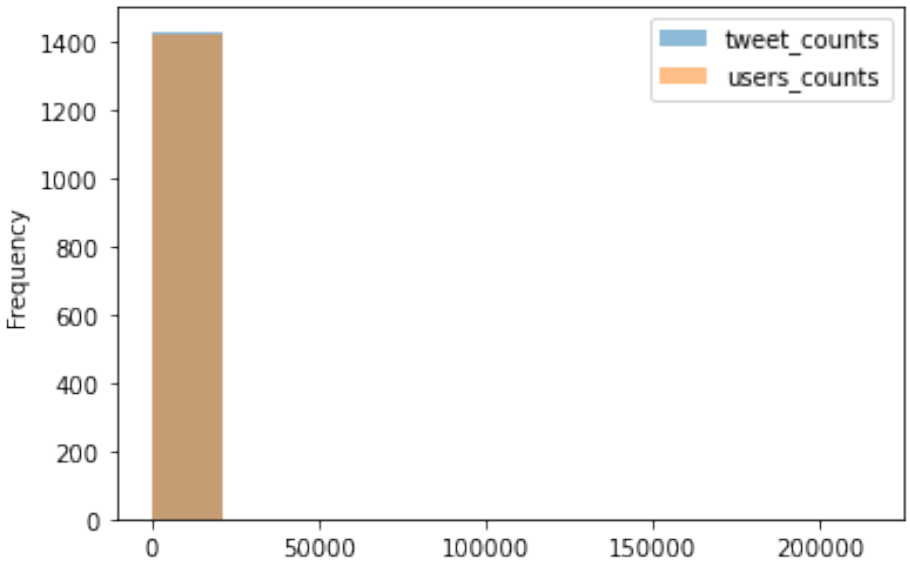
9. Plot the histogram based on the tweets counts

```
data_df_user_id_users_count.hist(column='tweet_counts',bins=20,
figsize=(12,8))
```



10. Plot the histogram based on the alpha 0.5

```
data_df_user_id_users_count.plot.hist(alpha=0.5)
```



Observation

1. In the point-7 graph it states the distribution between user_counts and tweet_counts. We can see the highest tweet count is 10503 tweeted by a single person.

2. In the point-8 graph shows the distribution of the tweet counts, in an average will get to know most of the user's average rate of tweeting is in a range of 500-550 per month.

3. In the point-9 give you a high glimpse of *users_count* and tweet counts where more tweets exists than users. The highest tweet is in the range of 1400.

4. The more tweets which are in a range of 500-550 which has been tweeted by most of the users.

5. There are very less users who have tweeted more than 1000.

6. When we segregate the dataset we will get to know there are less users and more tweets, we can also identify from which country the most tweets are coming which we will discuss in the later section.

7. To get the perfect distribution we are performing an aggregation on multiple levels, first level is on the tweets and second level on the users count.

8. Almost the distribution looks similar but the tweets are from different regions and there are very less users who have tweeted more than 1000 for June 2021 month.

9. We can see more segregated level data if we increase the bin size in the histogram.

10. The ratio what we see is the 1:2.

2. Find the top-5 users by total number of tweets. Do you think any are automated accounts (aka. bots)? Justify your answer.

Getting ready

In Twitter dataset we have a user's profile data where we can track his tweet . As part of this task, we need to find a top 5 users and need to identify whether are there any bots or are there a real users.

How to do it...

For this task we need to pull the fields like '*text*', '*user, 'id', 'country', 'place',' tweet time' and all other mentions so that it will help us to identify whether he is a genuine user*.

The following are the steps:

1. Iterate though the compression files and at the same time iterate over json files, load the dataset into a list and then create a pandas dataframe using that list.

2. Instead of reading the entire dataset set multiple times we can write the data into a csv.

3. Create a dataframe, there are chances where we get bad records while reading the file, please pass some additional parameters to resolve those issue as shown below

4. Now we have the dataframe with user name, user screen name, user id, place, country fields and the tweets.

5. Perform a grouping operation based on the user_id and get the count of his tweets

```
data_df_user_id_tweet_count_sort =
data_df_user_id_tweet_count.sort_values(by ='tweet_counts',
ascending=False)
data_df_user_id_tweet_count_sort =
data_df_user_id_tweet_count_sort.reset_index()

data_df_user_id_tweet_count_sort.head(6)
```

	index	user_id	tweet_counts
0	203067	1384110594	10503
1	93459	1211606433399095296	10002
2	696781	974188940973486081	8661
3	201860	1382496899744301057	8368
4	257374	161262801	8011

6. Here the user id 1384110594 has tweeted more when compare to others.

7. Let's pull out the details of these top 5 users

```
top_5_users_details = top_5_users.merge(data_df_copy,
on='user_id')
top_5_users_details.user_name.unique()
```

Output:

```
"L'hora catalana"
'Maria'
'casimiroperezcastrosin'
'Ayfer Güler'
'Radio TEDDY Playlist'
```

8. To validate whether they are real users are automated bots we need to an alyse their tweets and the timestamp of tweet, there are chances where it may be a scheduled cron bot.

Observation

1. As per the analysation I have observed these are the real users and not a bots.

2. As per the analysis I have checked each user's profile, followers count, re-tweeted counts, friends, and those looks genuine.

3. I have printed the timings for each user and they varies, if we thin they are bots then there will be more spams.

4. I have analysed in a deeper level and can state that these are the real users not a bots.

5. A Twitter bot is a bot software that uses the Twitter API to interact and engage with Twitter users. The Twitter bots can be programmed or automated to perform a specific task or series of tasks. It can autonomously tweet, retweet, like, follow, unfollow, or DM other accounts.

6. The top 5 bots of twitter are
   a. @BotSentinel
   b. @HundredZeros
   c. @MagicRealismBot
   d. @DearAssistant
   e. @Earthquakbot

7. To track a bot we can do the following
   a. Track Twitter follower growth
   b. Get instant notifications on gaining/losing each follower
   c. Identify and extract lists of fake followers/bots and inactive accounts
   d. Identify trolls, bots, and spam accounts

8. Twitter bots can be a great tool to provide useful information, interact with users on your behalf, or even assist in customer support. But Twitter bots pretending to be genuine users do not add value in any way other than boosting your follower count. Cleanse your Twitter follower-base of such accounts to get actual metrics that help in measuring the impact of your strategies with FollowerAudit.

3. Find the 5 users who receive the most mentions and comment on this.

Getting ready

In Twitter dataset we have a user's profile data where we can track his tweet . As part of this task, we need to find a top 5 users who have received the most mentions. Mention means the user names have been mentioned on other user tweets.

How to do it...

For this task we need to pull the fields like '*text*', '*user, 'id', 'country', 'place',' tweet time' and all other mentions so that it will help us to identify whether he is a top user*.

The following are the steps:

1. Iterate though the compression files and at the same time iterate over json files, load the dataset into a list and then create a pandas dataframe using that list.

2. Instead of reading the entire dataset set multiple times we can write the data into a csv.

3. Create a dataframe, there are chances where we get bad records while reading the file, please pass some additional parameters to resolve those issue as shown below.

4. Extract tweets and store it into a Series object.

```
data_df_copy_tweets = data_df_copy['text']
```

5. Load the Series dataset into a list

```
       json_list = []
for key, value in data_df_copy_tweets.iteritems():
    json_list.append(value)
```

6. Fetch the mentions by using a Regular Expressions

```
import re
data_search = '@([a-zA-Z0-9_.-?!^*$#/|,.{}]+)'
list_names=[]

for i in json_list:
    matched_value = re.findall(data_search, i)
    if matched_value == ' ' or matched_value == '' or
matched_value == None :
```

```
        pass
    else:
        list_names.append(matched_value)
```

7. There are some users who have tweeted multiple mentions in the same tweet, the below code helps us to make flatten the list

```
range_list=[]
for l in list_names:
    if(len(l) > 0) :
        #print(l)
        length = len(l)
        #print(length)
        if length >= 1:
            for row in l:
                #print(row)
                range_list.append(row)
```

8. Create a Dataframe usint the list range_list which have all the mentions.

9. Perform an aggregation on the user name which has been mentioned most of the times on the tweets

```
data_df_user_id_users_count =
data_df.mentioned_tweeet_user_names.value_counts()
data_df_user_id_users_count.columns = ['users_counts']
data_df_user_id_users_count =
data_df_user_id_users_count.reset_index()
```

```
data_df_user_id_users_count.sort_values(by=['mentioned_tweeet_u
ser_names'], ascending=False)
```

Output:

	user_name	mentioned_tweeet_user_names
0	YouTube	14416
1	MTV	10662
2	BorisJohnson	10652
3	dimash_official	10429
4	sedat_peker	10160

Observation

1. The toppest user who have received most mentions is *Youtube*, based on this we can understand how much people are talking about Youtube in twitter.

2. Rest of them are like MTV, Boris Jhonson, dimansh official, sedat pekar.

3. Dimash has an angelic voice and amazing range, earning him the nickname “The Six-Octave Man.” In fact, there are quite a few videos on YouTube featuring vocal coaches reacting to his performances. He has also performed in multiple languages, including French and English.

4. Reis Sedat Peker is a Turkish mafia boss and whistle-blower who has made various allegations about Turkish politicians and numerous government engagements in illegal activities through his own YouTube channel. He has described himself as a Pan-Turkist and Turanist.

5. MTV Live is an international 24-hour high-definition live music pay television network that is operated by Paramount International Networks' subsidiary Paramount Networks EMEAA. The channel is available in Europe, Latin America, Oceania, Middle East and North Africa and Asia

4. Choose 4 countries and compute how often they mention each other. This means you should compute 16 numbers e.g. UK mentions UK, UK mentions France, France mentions UK etc. Comment on any patterns you observe.

Getting ready

In Twitter dataset we have a *country* tag under *place* tag, where we can get the country of a user. In this task we need to choose any 4 countries and need to find out tweets talking about each others.

Totally there will be a 16 combinations and we need to analyse the tweets and need to provide our inputs.

Chosen countries:

1. *United Kingdom*
2. *France*
3. *Germany*
4. *Switzerland*

Patterns:

1.

United Kingdom → United Kingdom
United Kingdom → France
United Kingdom → Germany
United Kingdom → Switzerland

2.

France → France
France → United Kingdom
France → Germany
France → Switzerland

3.

Germany → Germany
Germany → United Kingdom
Germany → France
Germany → Switzerland

4.

Switzerland → Switzerland
Switzerland → United Kingdom
Switzerland → France
Switzerland → Germany

How to do it...

For this task we need to pull the fields like 'text', 'user, 'id', 'country', 'place', so that it will help us to observe any pattern between these countries.

The following are the steps:

1. Iterate though the compression files and at the same time iterate over json files, load the dataset into a list and then create a pandas dataframe using that list.

2. Instead of reading the entire dataset set multiple times we can write the data into a csv.

3. Create a dataframe, there are chances where we get bad records while reading the file, please pass some additional parameters to resolve those issue as shown below.

4. Create a dataframe based on the countries and fetch the records.

```
data_df_uk = data_df_copy[data_df_copy.country == 'United
Kingdom']
data_df_uk_uk =
data_df_uk[data_df_uk["text"].str.contains("United Kingdom")]
data_df_uk_france =
data_df_uk[data_df_uk["text"].str.contains("France")]
data_df_uk_germany =
data_df_uk[data_df_uk["text"].str.contains("Germany")]
data_df_uk_switzerland =
data_df_uk[data_df_uk["text"].str.contains("Switzerland")]
```

```
data_df_france = data_df_copy[data_df_copy.country == 'France']
data_df_france_uk =
data_df_france[data_df_france["text"].str.contains("United
Kingdom")]
data_df_france_france =
data_df_france[data_df_france["text"].str.contains("France")]
data_df_france_germany =
data_df_france[data_df_france["text"].str.contains("Germany")]
data_df_france_switzerland =
data_df_france[data_df_france["text"].str.contains("Switzerland
")]
```

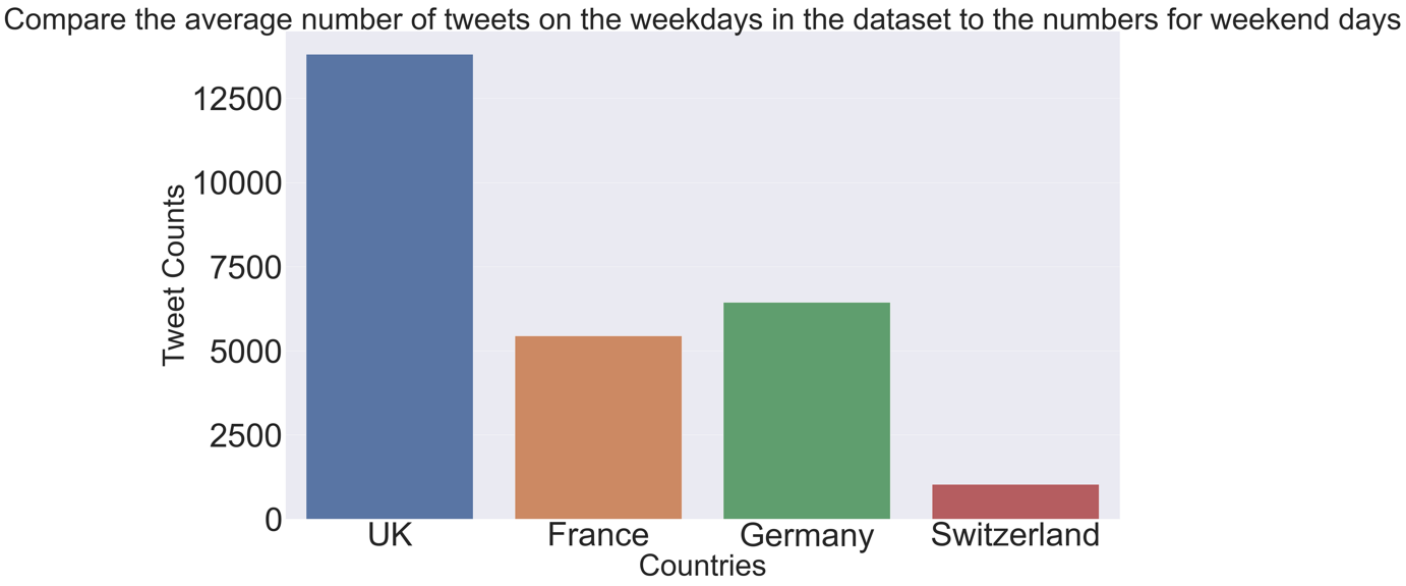
```
data_df_germany = data_df_copy[data_df_copy.country ==
'Germany']
data_df_germany_uk =
data_df_germany[data_df_germany["text"].str.contains("United
Kingdom")]
data_df_germany_france =
data_df_germany[data_df_germany["text"].str.contains("France")]
data_df_germany_germany =
data_df_germany[data_df_germany["text"].str.contains("Germany")
]
```

```
data_df_germany_switzerland =
data_df_germany[data_df_germany["text"].str.contains("Switzerla
nd")]
```

```
data_df_switzerland = data_df_copy[data_df_copy.country ==
'Switzerland']
data_df_switzerland_uk =
data_df_switzerland[data_df_switzerland["text"].str.contains("U
nited Kingdom")]
data_df_switzerland_france =
data_df_switzerland[data_df_switzerland["text"].str.contains("F
rance")]
data_df_switzerland_germany =
data_df_switzerland[data_df_switzerland["text"].str.contains("G
ermany")]
data_df_switzerland_switzerland =
data_df_switzerland[data_df_switzerland["text"].str.contains("S
witzerland")]
```

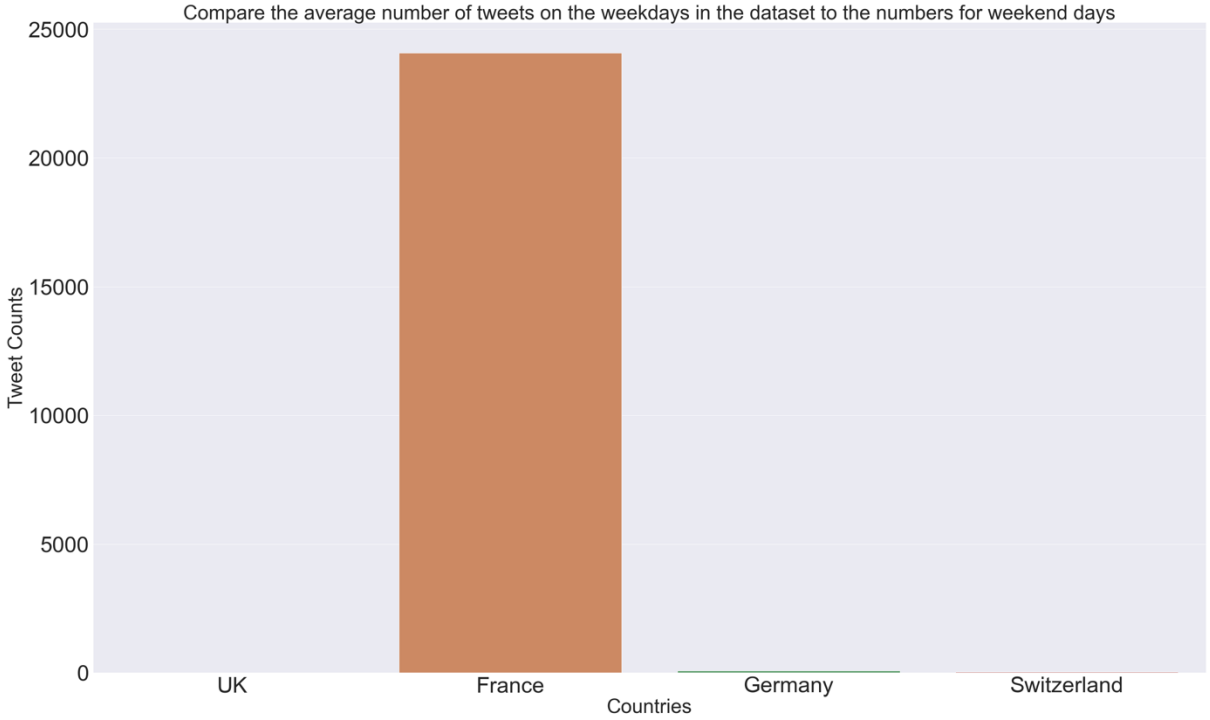
Observation

1. Plotting the bar diagram for the UK computation, with other three countries (France, Germany, Switzerland)



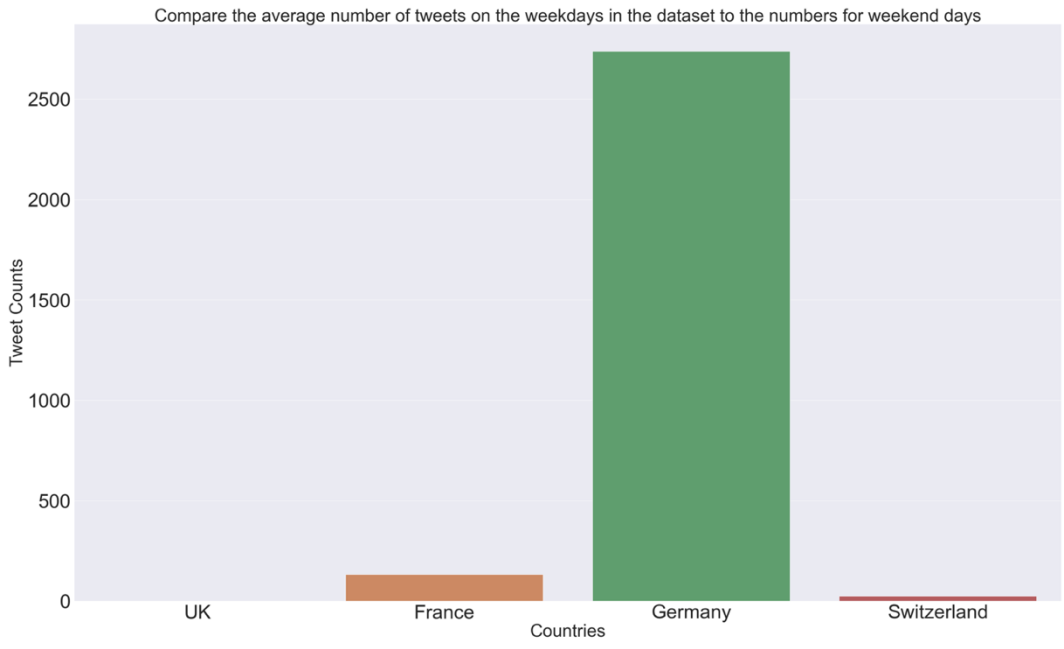
As per the above chart we can see UK more talk about its own country then Germany, France and Switzerland

2. Plotting the bar diagram for the France computation, with other three countries (UK, Germany, Switzerland)



As per the above chart we can see France more talk about its own country then Germany and Switzerland.

3. Plotting the bar diagram for the Germany computation, with other three countries (UK, France, Switzerland)



As per the above chart we can see Germany more talk about its own country then France and Switzerland.

4. Plotting the bar diagram for the Switzerland computation, with other three countries (UK, France, Germany)

As per the above chart we can see Switzerland more talk about its own country then France and Germany.

Part - 4
Events

1. Identify 3 days with unusually high activity in 3 different countries of your choosing. For example you could choose one day in the UK, one in France and one in Turkey. Describe and justify how you identify 'unusual' days.

Getting ready

In Twitter dataset we have to analyse the unusual high activity in three different countries based upon our selection.

Chosen Countries:

1. *United Kingdom*
2. *France*
3. *Germany*

How to do it...

For this task we need to pull the fields like 'text', 'user, 'id', 'country', 'place', so that it will help us to observe any pattern between these countries.

The following are the steps:

1. Iterate though the compression files and at the same time iterate over json files, load the dataset into a list and then create a pandas dataframe using that list.

2. Instead of reading the entire dataset set multiple times we can write the data into a csv.

3. Create a dataframe, there are chances where we get bad records while reading the file, please pass some additional parameters to resolve those issue as shown below.

4. Once we have the base dataset, we can split the dataset into three different datasets based on the countries.

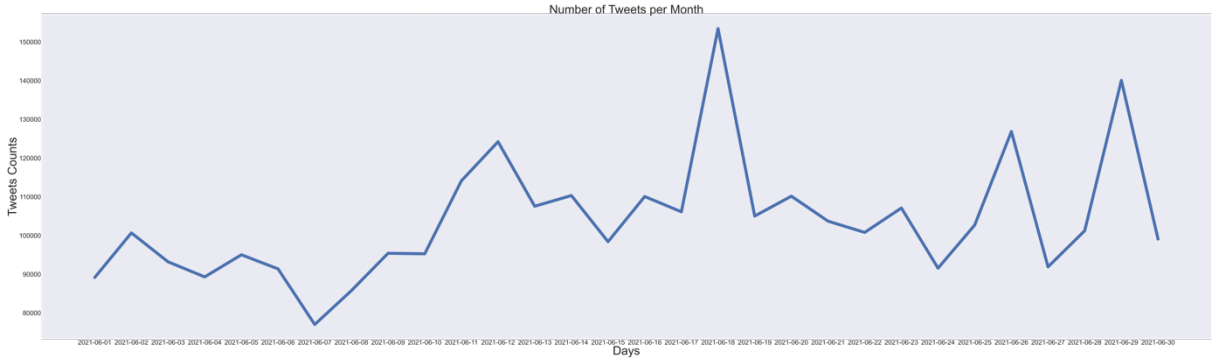
```
data_df_uk = data_df_copy[data_df_copy.country == 'United
Kingdom']
data_df_france = data_df_copy[data_df_copy.country == 'France']
data_df_germany = data_df_copy[data_df_copy.country ==
'Germany']
```

5. Starting with the UK dataset, let's try to perform the aggregation based on the tweet counts and try to find out which day has a high unusual activity.

```
data_df_uk_agg =
data_df_uk.groupby('date_data').agg({'text':['count']})
data_df_uk_agg.columns = ['tweet_counts']
data_df_uk_agg = data_df_uk_agg.reset_index()
```

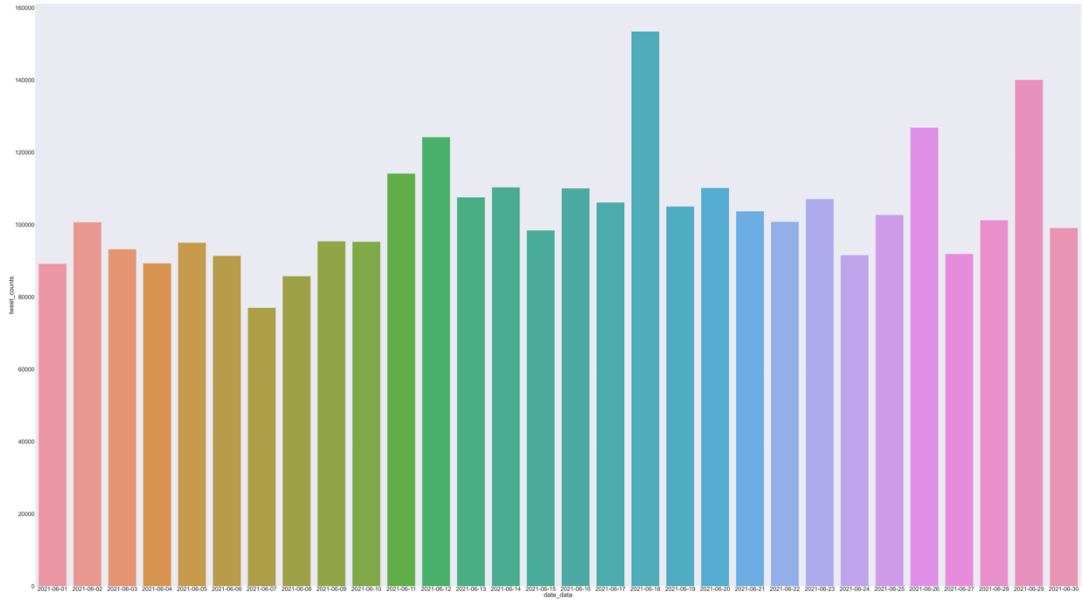
6. Plot the UK dataset to analyse which day we got most of the tweets.

```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize = (350,100))
plt.rcParams.update({'font.size': 100})
sns.set_style("whitegrid")
sns.set(font_scale = 10)
sns.lineplot(x='date_data', y='tweet_counts',
data=data_df_uk_agg, markers=True, dashes=True, linewidth=15,
err_style="bars")
plt.xlabel("Days",fontsize=200)
plt.ylabel("Tweets Counts",fontsize=200)
plt.title("Number of Tweets per Month",fontsize=200)
plt.show()
```



7. Plot the UK dataset using a bar diagram

```
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.axes as ax

plt.figure(figsize = (350,200))
plt.rcParams.update({'font.size': 100})
#sns.barplot(x='day_split', y='tweet_count', data=data_df_agg,
capsize = 0.05)
ax = sns.barplot(x='date_data', y='tweet_counts',
data=data_df_uk_agg, capsize = 0.05)
ax.bar_label(ax.containers[0])
plt.xlabel("Days",fontsize=200)
plt.ylabel("Tweets Counts",fontsize=200)
plt.title("Number of Tweets per Month",fontsize=200)
plt.show()
```

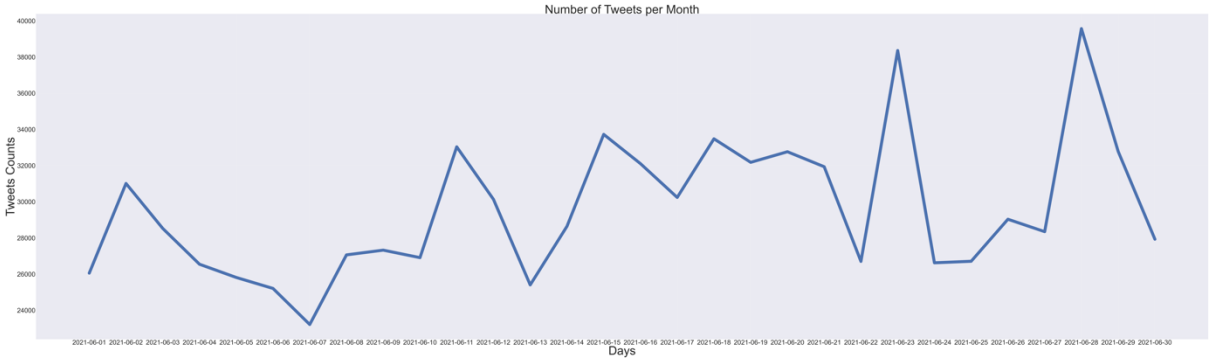


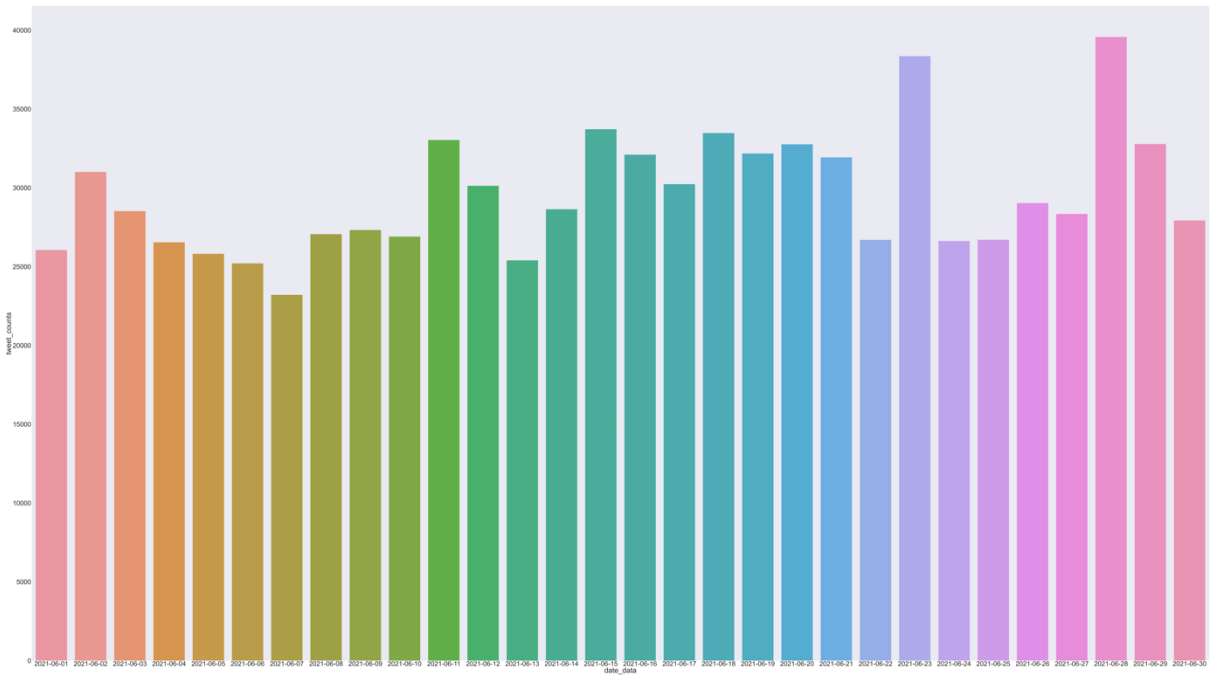
8. Perform an aggregation on other two countries i.e. France and Germany and perform an aggregation.

```
data_df_france_agg =
data_df_france.groupby('date_data').agg({'text':['count']})
data_df_france_agg.columns = ['tweet_counts']
data_df_france_agg = data_df_france_agg.reset_index()

data_df_germany_agg =
data_df_germany.groupby('date_data').agg({'text':['count']})
data_df_germany_agg.columns = ['tweet_counts']
data_df_germany_agg = data_df_germany_agg.reset_index()
```

9. Plot France dataset

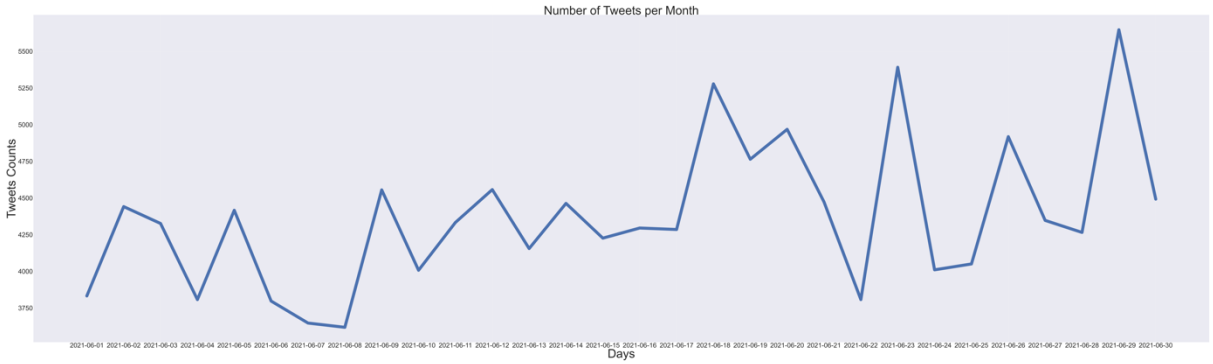
```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize = (350,100))
plt.rcParams.update({'font.size': 100})
sns.set_style("whitegrid")
sns.set(font_scale = 10)
sns.lineplot(x='date_data', y='tweet_counts',
data=data_df_france_agg, markers=True, dashes=True,
linewidth=15, err_style="bars")
plt.xlabel("Days",fontsize=200)
plt.ylabel("Tweets Counts",fontsize=200)
plt.title("Number of Tweets per Month",fontsize=200)
plt.show()
```

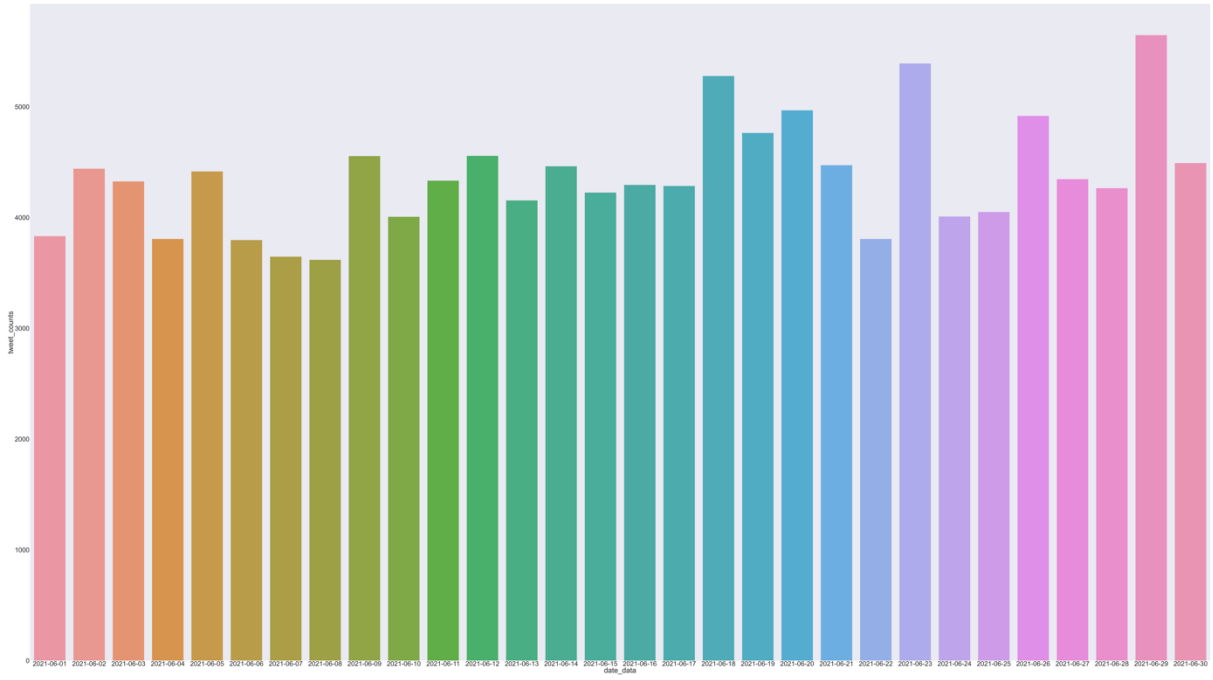




10. Plot Germany dataset

```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize = (350,100))
plt.rcParams.update({'font.size': 100})
sns.set_style("whitegrid")
sns.set(font_scale = 10)
sns.lineplot(x='date_data', y='tweet_counts',
data=data_df_germany_agg, markers=True, dashes=True,
linewidth=15, err_style="bars")
plt.xlabel("Days",fontsize=200)
plt.ylabel("Tweets Counts",fontsize=200)
plt.title("Number of Tweets per Month",fontsize=200)
plt.show()
```





Observation

1. The following are the dates with high unusual activity
   UK : 2021-06-18
   France : 2021-06-28
   Germany : 2021-06-29

2. In ***UK*** on the high unusual activity day people are more talking about

   - On 2021-06-18 there was a football match between England and Scotland EURO 2020 people were talking about it.
   - People were discussing about *Guterres re-elected for second-term as UN Secretary General*
   - People were discussing about *Serbia's Vucic hits back as Montenegro bans Srebrenica genocide denial*
   - 'Psychic' Thai lion predicts weekend's Euro games
   - Skunk water used against Palestinian protesters

3. In ***France*** on the high unusual activity day people are more talking about

   - The 2020 UEFA European Football Championship, commonly referred to as UEFA Euro 2020 or simply Euro 2020, was the 16th UEFA European Championship
   - The winner of France's elections was neither Macron nor Le Pen, but apathy
   - France has a new conservative leader who's looking to topple President Macron
   - Apple Daily to close, last pro-democracy Hong Kong newspaper
   - Bertrand also said Sunday that the result gave him the strength to meet the needs of all French people, hinting at his aspirations of becoming president at elections next year.

4. In ***Germany*** on the high unusual activity day people are more talking about

   - Germany turns rainbow-coloured in protest at UEFA stadium ban
   - German states seeing surge in Delta variant Covid cases
   - Ancient sculptures prompt Germany to reckon with colonial past
   - Merkel condemns Hungary's LGBTQ law as 'wrong'
   - German word of the day: Das Tanzverbot

2. Characterise each of these three days by

a. Displaying some indicative Tweets.

Getting ready

As in our previous section we have identified the unusual high activity days, try to get the tweets and identify the patterns.

How to do it...

We need to fetch the tweets by applying the date filter, the date should be the high unusual date based on the country.

```
data_df_uk_high_tweets = data_df_uk[data_df_uk.date_data ==
'2021-06-18']
data_df_france_high_tweets =
data_df_france[data_df_france.date_data == '2021-06-28']
data_df_germany_high_tweets =
data_df_germany[data_df_germany.date_data == '2021-06-29']
```

Observation

Tweets:

UK

- The EURO 2020 Football Village is closed today.
- António Guterres secures second term as UN Secretary-General, calls for new era of 'solidarity and equality'
- @SerbianFooty Oooh the burn 😂
- Serbia and Bosnia... One day ❤️❤️, 🇷🇸🇧🇦
- England vs Scotland game prediction 4-1 by the lion
- Last Euros supporting Wales I was on tour driv.
- @citizen_tammy @BrianSpanner1 Is it the Skunk?

France

- @maxencel_28 @GTX040204020402 @Kingsley_973
- @bendjelel @addhislam @FeizaBM Ou encore les
- Les sondages vs la réalité #electionsregionale..
- Excellent récap du second tour des
- Ma Femme qui boude depuis 2h dans la chambre
- И все же M1 одно удовольствие 👌#Apple

Germany

- Die #EURO2020 darf nicht das neue #Ischgl werd.
- @ThomasWalde Die #EURO2020 darf nicht das neue
- A Midsummer Night's Nightmare" #DeltaVariant
- Merkel tolong naikin Steuer buat Fußballspiel.
- @Dastanzverbot Ist leider Schrott. Geld nicht wert

b. Making a word cloud from the tweet text.

Getting ready

In this section we need to display the word cloud using a wordcloud api, these are the words which has been tweeted most of the time on that specified unusual high activity day.

How to do it...

1. We need to install wordcloud package using conda/pip3.
2. We need to come up with the stopwords, so that we can exclude those.
3. Fetch the dataset and load it into dataframe separated by the countries.

```
data_df_uk = data_df[data_df_copy.country == 'United Kingdom']
data_df_france = data_df[data_df_copy.country == 'France']
data_df_germany = data_df[data_df_copy.country == 'Germany']
```

4. Extract the tweets of UK dataset.

```
text = " ".join(review for review in
data_df_uk.text.astype(str))
```

5. Set the Stopwords

```
stopwords = set(STOPWORDS)

stopwords.update(["bloom", "rose", "petals", "Average",
"diameter","flushes", 'throughout', "Blooms", "form", "https",
"Hi", "us", 'where', 'done', 'if', 'before', 'll', 'very',
'keep', 'something', 'nothing', 'thereupon',
    'may', 'why', 'â€™s', 'therefore', 'you', 'with',
'towards', 'make', 'really', 'few', 'former',
    'during', 'mine', 'do', 'would', 'of', 'off', 'six',
'yourself', 'becoming', 'through', 'Thank',
    'seeming', 'hence', 'us'])
```

6. Generate word cloud text

```
wordcloud = WordCloud(stopwords=stopwords,
background_color="white", width=800, height=400).generate(text)
plt.axis("off")
plt.figure(figsize=(30,10))
plt.tight_layout(pad=0)
plt.imshow(wordcloud, interpolation='bilinear')
plt.show()
```

7. We can also use the *mask* functionality of wordcloud to print the wordcloud into different map patterns.

```
from PIL import Image
rose_mask = np.array(Image.open("1599661.jpeg"))
germany_mask = np.array(Image.open("germany_map.jpeg"))
france_mask = np.array(Image.open("france_map.jpeg"))
uk_mask = np.array(Image.open("uk_map.jpeg"))
```

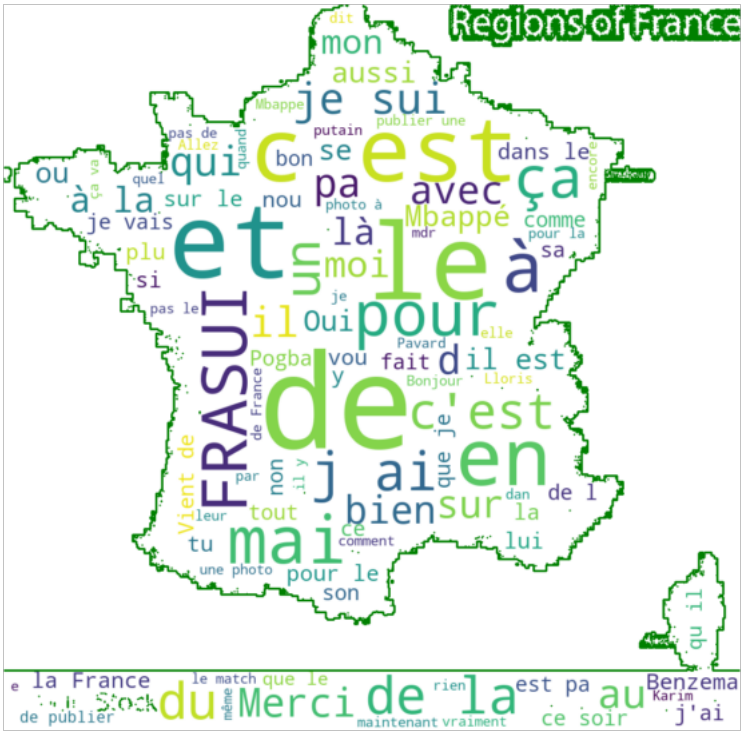
8. Here we are using numpy array method which reads a jpeg/png files and convert into a multidimensional array.

```
wc = WordCloud(background_color="white", max_words=100,
mask=rose_mask,
               stopwords=stopwords, contour_width=3,
contour_color='green')

# Generate a wordcloud

wc.generate(text)

# store to file

wc.to_file("rose_bloom.png")

# show
plt.figure(figsize=[20,10])
plt.imshow(wc, interpolation='bilinear')
plt.axis("off")
plt.show()
```

9. Plot UK word cloud data into a UK map

```
wc = WordCloud(background_color="white", max_words=100,
mask=uk_mask,
               stopwords=stopwords, contour_width=4,
contour_color='green')

# Generate a wordcloud

wc.generate(text)

# store to file

wc.to_file("rose_bloom.png")

# show

plt.figure(figsize=[20,10])
plt.imshow(wc, interpolation='bilinear')
plt.axis("off")
plt.show()
```

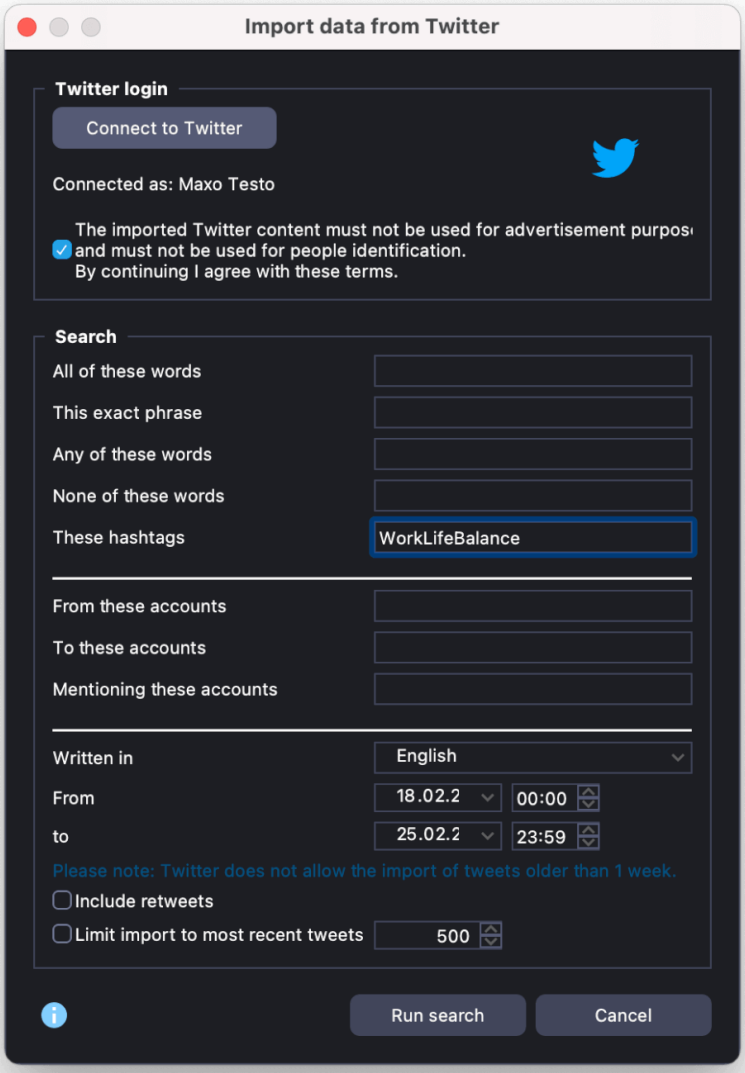
10. Plot France wordcloud data into France Map



11. Plot Germany wordcloud data into Germany Map

c. Any other method.

We can analyse our tweets by various methods for example we can use MAXQDA tool for our analysis where we can perform stages of social media analysis.



After performing an analysis got the similar results as like the previous one.


let
people
sure
team
made
say
work
lol
still
Sterling
ENGSCO
love
will
amp
got
first
win
morning
though
thing
great
going
better
best
brilliant
go
one
now
well
give
want
Thanks
start
even
m
posted
Grealish
come
know
way
tonight
always
back
please
thought
getting
Oh
amazing
d
point
Scotland
watch
reyear
look
Southgate
today
put
never
time
hope
absolutely
right
lovely
far
Ye
photo
play
England
week
feel
watching
playing
Happy
fucking
nice
said
bit
football
new
x
see
game
big
lot
think
take
man
good
day
player
need
ve
many
mate
Kane
much
shit
don
fuck

3. Summarise the events you have detected and validate with some other source of data e.g. news articles.

1. The events which got detected with high unusual activity for the three countries UK, France, Germany are matching with the articles which got published in newspapers for example

UK: For the high unusual activity date 2021-06-18 the news is published as

- England v Scotland: Thousands of fans descend on London for goalless draw.
- Guterres re-elected for second-term as UN Secretary General
- António Guterres secures second term as UN Secretary-General, calls for new era of 'solidarity and equality'
- Serbia's Vucic hits back as Montenegro bans Srebrenica genocide denial
- Move over Paul the Octopus, Thailand has a new claimant to the football forecasting crown: Boy the "psychic" lion, who has so far correctly called four Euro 2020 matches.
- Skunk water used against Palestinian protesters

France: For the high unusual activity date 2021-06-28 the news is published as

- UEFA EURO 2020 will be the biggest celebration of football in the tournament's history, and we're honoured to play a part in hosting it.
- The winner of France's elections was neither Macron nor Le Pen, but apathy.
- France has a new conservative leader who's looking to topple President Macron
- Apple Daily: Hong Kong pro-democracy paper announces closure.

Germany: For the high unusual activity date 2021-06-29 the news is published as

- Germany turns rainbow-coloured in protest at UEFA stadium ban
- 'Vaccinate quickly': German states seeing surge in Delta variant Covid cases
- Ancient sculptures prompt Germany to reckon with colonial past
- German word of the day: Das Tanzverbot
- Merkel Condemns Hungary's LGBTQ Law As 'Wrong'

Part - 5
Reflection

Using social media to study the real world is very common in academia, media and industry. Now that you have some experience analysing Twitter data discuss

•The use of Twitter to study the effectiveness of lockdown policies

Write no more than 500 words. Remember, this is an academic writing exercise. You should be citing sources and justifying your opinions with evidence/analysis.

Background

The emergence of SARS-CoV-2 (ie, COVID-19) has given rise to a global pandemic affecting 215 countries and over 40 million people as of October 2020. Meanwhile, we are also experiencing an infodemic induced by the overabundance of information.

Objective

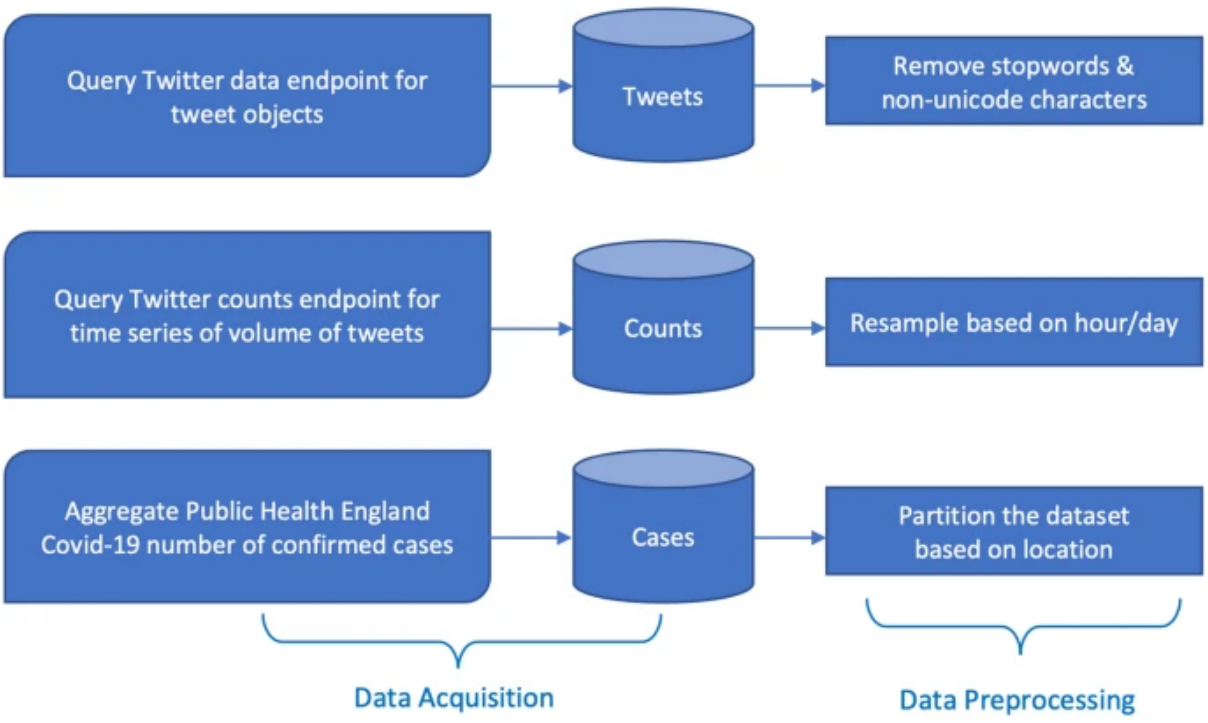
This study aimed to investigate COVID-19-related health beliefs on one of the mainstream social media platforms, Twitter, as well as potential impacting factors associated with fluctuations in health beliefs on social media.

Methods

We used COVID-19-related posts from the mainstream social media platform Twitter to monitor health beliefs. A total of 92,687,660 tweets corresponding to 8,967,986 unique users from January 6 to June 21, 2020, were retrieved. To quantify health beliefs, we employed the health belief model (HBM) with four core constructs: perceived susceptibility, perceived severity, perceived benefits, and perceived barriers.

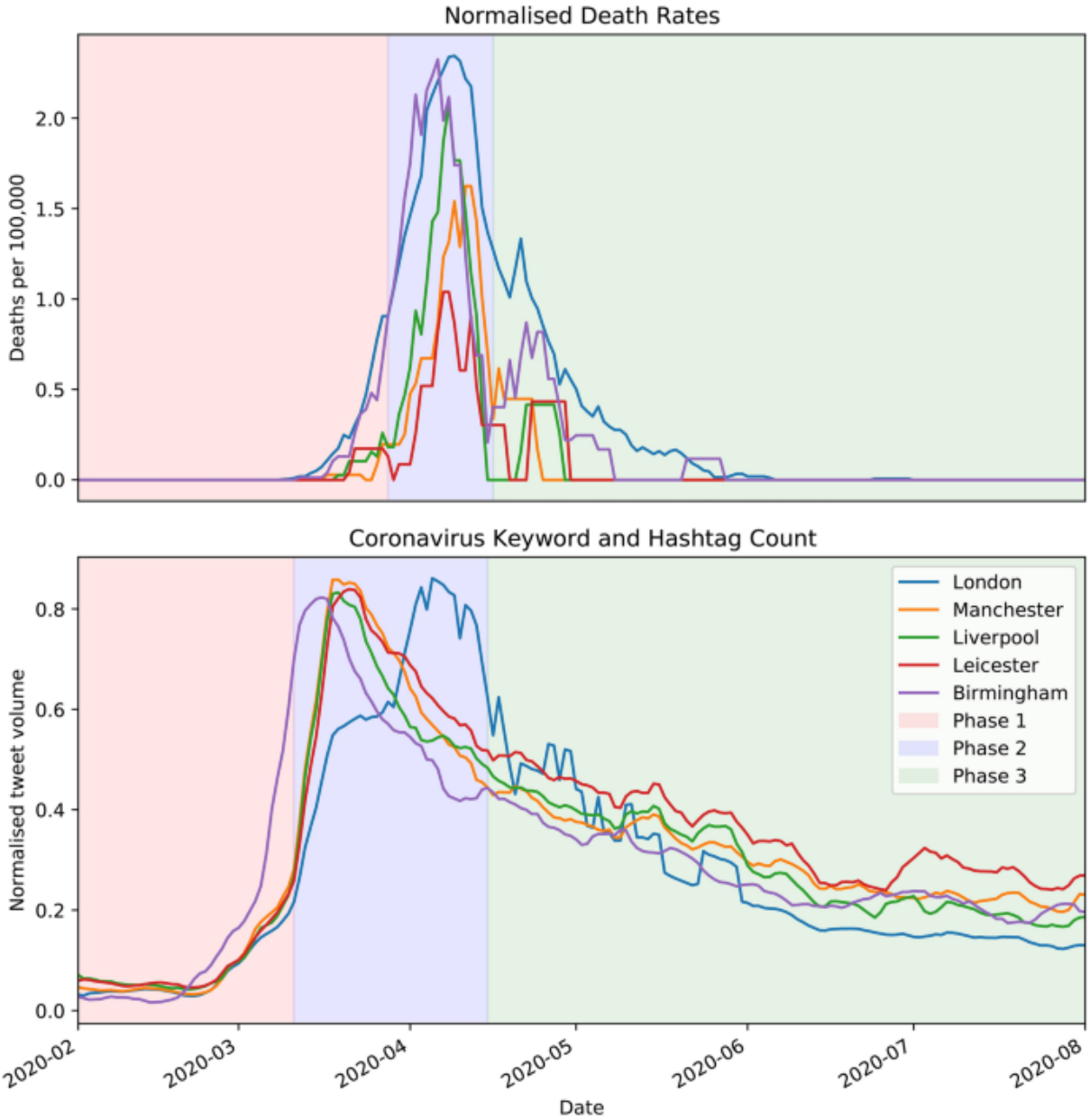
Material

A comprehensive COVID-19 Twitter data set containing all tweets from the UK over the duration of the pandemic would be ideal for performing outbreak prediction and behavioural analysis. However, due to limitations on the amount of data that could be collected via the Twitter API endpoints, a focused study on different cities in England was conducted. An initial exploratory analysis found that the cities with the largest fraction of tweets in the UK were London.

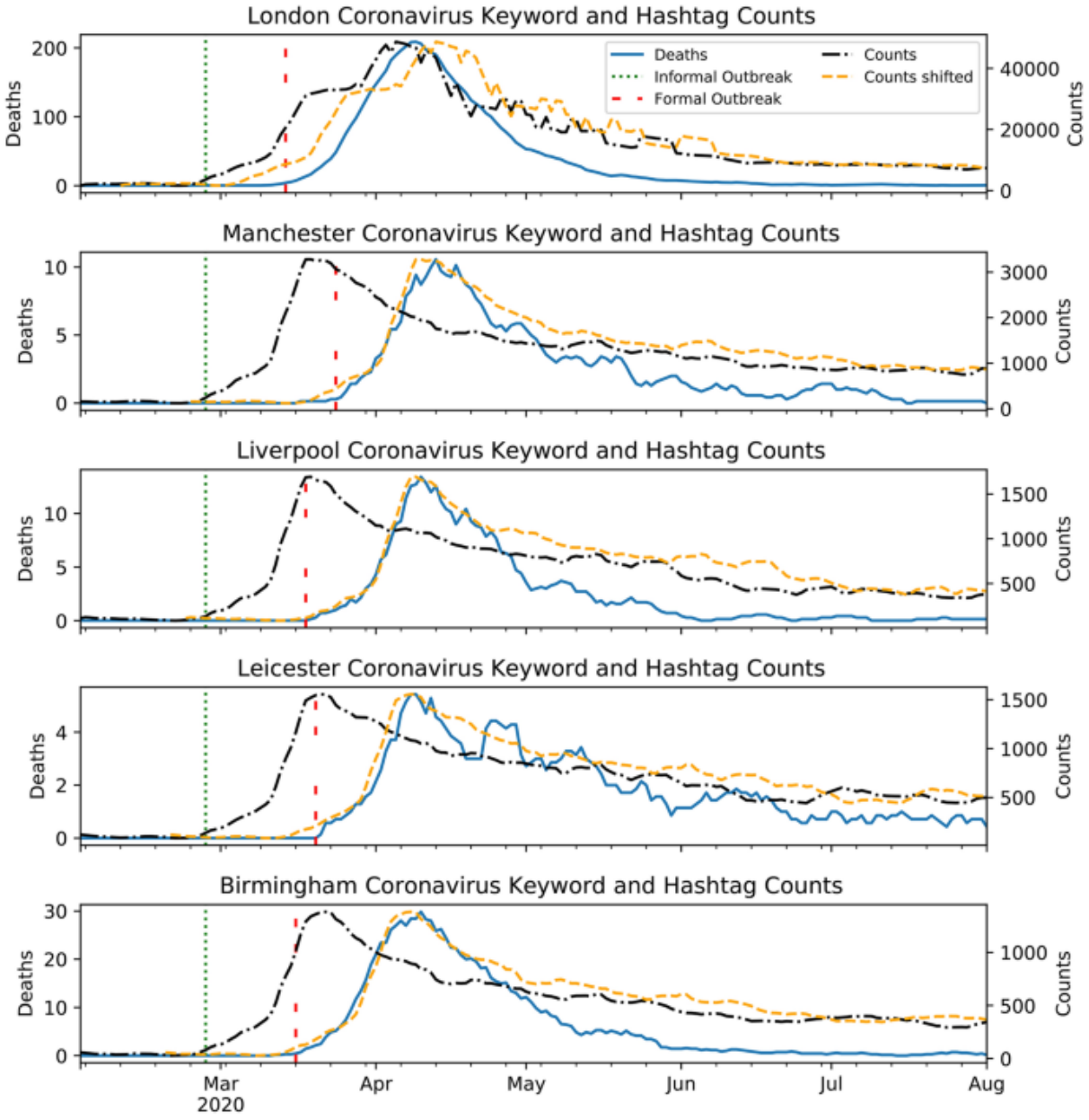


Results

The use of social media data to look at the prevalence of a disease has been considered in the past for avian influenza , and has also been considered for coronavirus. This phenomenon of Twitter's ability to provide insights into pandemic activity has been referred to as "wisdom of the crowds"; the collective knowledge of individual users2. By making use of this collective knowledge, the objective is to try and predict pandemic activity in the absence of any virus tracking system. In order to build an early-warning system, the total volume of tweets obtained via the counts endpoint was utilised within a time-series comparison with Public Health England (PHE) datasets on COVID-19 deaths. Using deaths as a means of tracking the outbreak was more meaningful than using confirmed COVID-19 cases.



Plots of the time series of the counts endpoint for ‘coronavirus’ or ‘#coronavirus’ or ‘covid’ or ‘#covid’ for five UK cities as well as death numbers between February and August 2020. Also plotted is the former shifted by the time lag that maximised the correlation coefficient. The dates of the formal and informal outbreaks are plotted as red and green vertical lines, respectively.



Word clouds of the corresponding positive (left) and negative (right) mask related tweets from Vader sentiment analysis. We observe rather distinct types of words in each, particularly more words of gratitude in the positive word cloud as expected.

Conclusion

As an analogy of the classic epidemiology model where an infection is considered to be spreading in a population with an *R*0 greater than 1, we found that the number of users tweeting about COVID-19 health beliefs was amplifying in an epidemic manner and could partially intensify the infodemic.

Keywords

COVID-19, social media, health belief, Twitter, infodemic, infodemiology, machine learning, natural language processing