CST 4050 – MODELLING, REGRESSION AND MACHINE LEARNING

Edison Mataj - M00732762

Abstract

The following report is based on the Machine Learning Module (4050) of the MSc Data Science Programme, that required us to come up with a machine learning problem and investigate and implement machine learning algorithms on how to tackle the issue.

DESCRIPTION OF THE DATASET AND THE MACHINE LEARNING CHALLENGE

Lately, we have been facing a pandemic virus outbreak that has changed our everyday life drastically. There are numerous datasets available regarding the number of cases in the whole world, the number of deaths, the number of recovered cases and different visualisations regarding the virus. However, there is still a lot of work to do in order to fight this virus, but given the fact that machine learning and big data are the two twin engines of AI, *will they help with this fight against the pandemic and is it able to make predictions based on actual datasets*?

1) How does the public react to the outbreak on social media?

For this part of the project I decided to scrape data from twitter based on their tweets about the keyword “coronavirus”. However, there were some limitations as twitter only allows a certain amount of data to be scraped for a certain time period and the computation power was a setback too. The project was constructed in Python and below I will go through the necessary steps to gather the data from twitter social media.

EXTRACTING THE DATA FROM TWITTER

```
pip install tweepy
```

```
Requirement already satisfied: tweepy in c:\users\user\anaconda3\lib\site-packages (3.8.0)
Requirement already satisfied: PySocks>=1.5.7 in c:\users\user\anaconda3\lib\site-packages (from tweepy) (1.7.1)
Requirement already satisfied: six>=1.10.0 in c:\users\user\anaconda3\lib\site-packages (from tweepy) (1.12.0)
Requirement already satisfied: requests-oauthlib>=0.7.0 in c:\users\user\anaconda3\lib\site-packages (from tweepy) (1.3.0)
Requirement already satisfied: requests>=2.11.1 in c:\users\user\anaconda3\lib\site-packages (from tweepy) (2.22.0)
Requirement already satisfied: oauthlib>=3.0.0 in c:\users\user\anaconda3\lib\site-packages (from requests-oauthlib>=0.7.0->twe
epy) (3.1.0)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in c:\users\user\anaconda3\lib\site-packages (from reque
sts>=2.11.1->tweepy) (1.24.2)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\user\anaconda3\lib\site-packages (from requests>=2.11.1->tweepy)
(2019.9.11)
Requirement already satisfied: idna<2.9,>=2.5 in c:\users\user\anaconda3\lib\site-packages (from requests>=2.11.1->tweepy) (2.
8)
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in c:\users\user\anaconda3\lib\site-packages (from requests>=2.11.1->tweep
y) (3.0.4)
Note: you may need to restart the kernel to use updated packages.
```

```
import os
import tweepy as tw
```

Figure 1. Tweepy package

As shown in the figure 1 above the first step was to install the “Tweepy” package on python, which is a package that allows users to make use of the Twitter API and read data from twitter.

The next step is setting the 4 properties that allow the connection with the twitter API as shown in the figure below:

```
accesstoken=('126915993-nBy3nfdP95tPsKdQG23vfgdkpLjGSLYcsMc4FK2p') #Assigning the API keys from Twitter and
accesstokensecret=('WKHZfUjiQFriffM7mwaFI92fEGtoPnd7Mov6Y6kw7xIGq')#setting the 4 properties
apikey=('9ex4RCd3eMnOLUIjR5VGqOZL6')
apisecretkey=('Hua9ES9JTuBKb8OFto0w06UxRiEtlSs5dUVVPSbuBDqcCqSkOk')
```

Figure 2. Assigning the 4 properties

To get the properties as shown above, first I had to create a Twitter developer account and create and app, which then would provide me with the Access token, Access token secret, API key and the API secret key.

The next step would be using the “tweepy” package to call the OAuth Handler, which is an authorization handler that requires passing the 4 properties.

```
auth = tw.OAuthHandler(apikey, apisecretkey) #Use the tweepy package and call OAuthHandler
auth.set_access_token(accesstoken, accesstokensecret) #It is an authorization handler to pass the API keys
api = tw.API(auth, wait_on_rate_limit=True)
```

Figure 3. OAuth Handler

As you can see above, there is also a wait on rate limit, which refers to twitter no allowing users to bombard their network at any time so there is a limit in which you can scrape data. This is set to approximately 15000 tweets per 15 minutes. In case a user exhausts the number of tweets then, the user must wait for 15 minutes in order to scrape more data. This would also be able to run in a loop overnight in order to get more data but given the computation power this was not possible.

The next step is to search for the keyword which in this case is “coronavirus”. Basically, every tweet containing the #coronavirus would be scraped from twitter.

```
search_words = "#coronavirus" #Assign the word we want to search for to the variable
date_since = "2020-04-01" #Assign the date from which we want to get the data
```

```
tweets = tw.Cursor(api.search,  #Goes to the TweepyCursor and applys the search for keyword coronavirus from the given date
                   q=search_words,
                   lang="en",
                   since=date_since).items(10000)
```

```
tweets
```

```
<tweepy.cursor.ItemIterator at 0x1926d58f908>
```

```
tweet_details = [[tweet.geo, tweet.text, tweet.user.screen_name, tweet.user.location] for tweet in tweets]
#tweet_details are pulled from twitter as mentioned above
```

Figure 4. Making the request

The last block of the figure above, allows us to change the tweet details that we want to get. In this example I was interested in getting the Geolocation (Lat/Lon), Text of the tweet, UserName and the UserLocation. However, in many cases the user location is private, hence not allowing us to get the location for a specific tweet.

DATA PREPROCESSING

The next step is converting all the data from twitter into a Pandas Dataframe and getting 4 columns as shown in the figure below:

```
import pandas as pd
tweet_df = pd.DataFrame(data=tweet_details, columns=['geo', 'text', 'user', 'location']) #creates the dataframe with the tweet d
```

```
pd.set_option('max_colwidth', 800)
tweet_df.head(5)
```

Figure 5. Converting the data into a Pandas Dataframe

In the figure below we get the first 5 records of our data.

```
pd.set_option('max_colwidth', 800)
tweet_df.head(5)
```

	geo	text	user	location
0	None	Fr. Clifford Mulasikwanda, a priest in #Zambia, found himself with an empty Church, a cell phone, and a second-hand tripod…	TracyOakley819	Dartmouth, Nova Scotia
1	None	UN says global hunger could double due to #coronavirus blow, as WHO says to prepare for a 'new way of living' amid the pande…	NgoiePado	
2	None	While in Coronavirus Lockdown, opens up about her plans of meeting Bigg Boss 13 bestie . Read On.…	asimcupcakes	
3	None	#selfisolationhelp Tip 36: If you're in #SelfIsolation / #Isolation due to #coronavirus Play cards! Patience can be…	HeatherJShort	
4	None	😉Wanting to know more about the #coronavirus and the impact on #genderequality? We have prepared a special webpage that exp…	alberto_soccol	

Figure 6. Displaying the first 5 records

Furthermore, we can display the user details and the locations as shown in Figure 7 below:

```
tweet_df.user.value_counts()
```

```
himmoderator      36
Arjuna3109        21
ThirupathiINC     14
PPersnickety      12
sumanebot         11
                  ..
The_Outlaw___      1
CommuterMetro1     1
AhoFrank           1
beesandbalm        1
mdiethert          1
Name: user, Length: 8524, dtype: int64
```

```
tweet_df.location.value_counts()
```

```
                         2885
India                     158
London                    114
United States             108
London, England           108
                         ...
#WesternMassVoter           1
Denmark                     1
ohio                        1
République du Rwanda        1
naas                        1
Name: location, Length: 3407, dtype: int64
```

Figure 7. Displaying user tweets and location

When we get the location, we can observe that many locations are shown in a city level, meaning that we will not get much insights when we aggregate. For instance, there is London and London, England as 2 separate locations, but by making use of the Google maps API we can group them by country level.

Moreover, the twitter data needs to be cleaned because if we observe there is the username, there is http, url and special characters, as part of the text of the tweet, which we are not interested in. As we can observe below that is how we can clean these special characters and urls:

```
import re
def clean_tweets(text): #function cleaning the dataset, more specifically the text part of the tweets
    text = re.sub("RT @[\w]*:", "", text)
    text = re.sub("@[\w]*", "", text)
    text = re.sub("https?://[A-Za-z0-9./]*", "", text)
    text = re.sub("\n", "", text)
    return text
```

```
tweet_df['text'] = tweet_df['text'].apply(lambda x: clean_tweets(x)) #applying the function above on the dataset
```

```
tweet_df.head(20)
```

	geo	text	user	location
0	None	Fr. Clifford Mulasikwanda, a priest in #Zambia, found himself with an empty Church, a cell phone, and a second-hand tripod...	TracyOakley819	Dartmouth, Nova Scotia
1	None	UN says global hunger could double due to #coronavirus blow, as WHO says to prepare for a 'new way of living' amid the pande...	NgoiePado	
2	None	While in Coronavirus Lockdown, opens up about her plans of meeting Bigg Boss 13 bestie . Read On....	asimcupcakes	
3	None	#selfisolationhelp Tip 36: If you're in #SelfIsolation / #Isolation due to #coronavirus Play cards! Patience can be...	HeatherJShort	
4	None	😉Wanting to know more about the #coronavirus and the impact on #genderequality? We have prepared a special webpage that exp...	alberto_soccol	
5	None	Question 2 as part of the "Be COVID Wise" series is here.How long does it take before one shows the symptoms of #Covid-1...	ScienceBagels	New Delhi, India

Figure 8. Cleaning the text of the tweets

From the figure above we can observe that first, I created a “clean_tweets” function to clean the twitter data and then I applied that function as a “Lambda function” in all the data. Now we can see that most of the special characters are cleaned and there is no username or urls inside the text part of the data.

The next step is saving the dataframe into a CSV file that we are going to use for further processing. Then I installed the “spacy” package, which allows us to make use of the “en_core_web_sm” library. This library breaks down all the text from the twitter data and assigns the most used words into different entities, such as: GPE that stands for Geopolitical Entity, DATE that stands for the date, CARDINAL that stands for the numbers, ORG that stands for organizations, etc.

```
import spacy
```

```
import en_core_web_sm
nlp = en_core_web_sm.load()
```

```
tweet_df['text'].apply(lambda x: [print("\tText: {}, Entity : {}".format(ent.text, ent.label_)) if (not ent.text.startswit
#this function gets the entities and prints them for the tweets
```

```
        Text: 2 days, Entity : DATE
        Text: Edgbaston Cricke, Entity : PERSON
        Text: Indonesia, Entity : GPE
        Text: Iran, Entity : GPE
        Text: Coronavirus, Entity : MONEY
        Text: the TRUE &amp, Entity : ORG
        Text: CCP, Entity : ORG
        Text: CHINESE Communitist Party, Entity : ORG
        Text: CoronaVirus, Entity : ORG
        Text: Indians, Entity : NORP
        Text: UK, Entity : GPE
        Text: Coronavirus, Entity : MONEY
        Text: week ending, Entity : DATE
        Text: April 2020, Entity : DATE
        Text: 6,213, Entity : CARDINAL
        Text: Italy, Entity : GPE

        Text: the end of this week, Entity : DATE
        Text: daily, Entity : DATE
        Text: Ratnagiri, Entity : MONEY
```

Figure 9. Entities of the tweets

Next, I created a new column called “entities” and applied the function above to fill in all the values for the entities based on the tweets in the new column that I created.

```
tweet_df['entities'] = tweet_df['text'].apply(lambda x: [(ent.text, ent.label_) if (not ent.text.startswith('#')) else "" for en
#creates a new column called entities on the pandas dataframe with the entities part extracted from the function created above
```

```
tweet_df.head(20)
```

	geo	text	user	location	entities
0	None	Fr. Clifford Mulasikwanda, a priest in #Zambia, found himself with an empty Church, a cell phone, and a second-hand tripod...	TracyOakley819	Dartmouth, Nova Scotia	[(Clifford Mulasikwanda, PERSON), , (second, ORDINAL)]
1	None	UN says global hunger could double due to #coronavirus blow, as WHO says to prepare for a 'new way of living' amid the pande...	NgoiePado		[(UN, ORG), (WHO, ORG)]
2	None	While in Coronavirus Lockdown, opens up about her plans of meeting Bigg Boss 13 bestie . Read On....	asimcupcakes		[(Coronavirus Lockdown, PERSON), (Bigg Boss, FAC), (13, CARDINAL)]
3	None	#selfisolationhelp Tip 36: If you're in #SelfIsolation / #Isolation due to #coronavirus Play cards! Patience can be...	HeatherJShort		[, (36, CARDINAL), ,]
4	None	Wanting to know more about the #coronavirus and the impact on #genderequality? We have prepared a special webpage that exp...	alberto_soccol		[]
5	None	Question 2 as part of the "Be COVID Wise" series is here.How long does it take before one shows the symptoms of #Covid-1...	ScienceBagels	New Delhi, India	[(2, CARDINAL), (one, CARDINAL),]

Figure 10. Data with the new column “entities”

Furthermore, I decided to apply a sentiment analyser called the "vader_lexicon", which is part of the "nlt" package in python. This is simply a rules-based analyser which is tested on social media in order to get useful insights from it.

```
import nltk #using nltk package and the vader sentiment analyzer
from nltk.sentiment.vader import SentimentIntensityAnalyzer #it is a rules based analyzer
nltk.download('vader_lexicon')
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     C:\Users\user\AppData\Roaming\nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
```

```
True
```

```
sid = SentimentIntensityAnalyzer() #creating the sentiment analyzer object
```

```
tweet_df['sentiment'] = tweet_df['text'].apply(lambda x: sid.polarity_scores(x)) #creating another column called sentiment
#calling the sentiment analyzer object created above to get the polarity scores for the text of the tweets
```

Figure 11. Applying the sentiment analyser

Next, I used the function above to assign all the polarity scores (Positive, Negative, Neutral, Compound) in a new column called "sentiment", as shown on the figure below.

	geo	text	user	location	entities	sentiment
0	None	Fr. Clifford Mulasikwanda, a priest in #Zambia, found himself with an empty Church, a cell phone, and a second-hand tripod...	TracyOakley819	Dartmouth, Nova Scotia	[(Clifford Mulasikwanda, PERSON), , (second, ORDINAL)]	{'neg': 0.101, 'neu': 0.899, 'pos': 0.0, 'compound': -0.2023}
1	None	UN says global hunger could double due to #coronavirus blow, as WHO says to prepare for a 'new way of living' amid the pande...	NgoiePado		[(UN, ORG), (WHO, ORG)]	{'neg': 0.083, 'neu': 0.917, 'pos': 0.0, 'compound': -0.25}
2	None	While in Coronavirus Lockdown, opens up about her plans of meeting Bigg Boss 13 bestie . Read On....	asimcupcakes		[(Coronavirus Lockdown, PERSON), (Bigg Boss, FAC), (13, CARDINAL)]	{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
3	None	#selfisolationhelp Tip 36: If you're in #SelfIsolation / #Isolation due to #coronavirus Play cards! Patience can be...	HeatherJShort		[, (36, CARDINAL), ,]	{'neg': 0.0, 'neu': 0.848, 'pos': 0.152, 'compound': 0.4003}
4	None	😊Wanting to know more about the #coronavirus and the impact on #genderequality? We have prepared a special webpage that exp...	alberto_soccol		[]	{'neg': 0.0, 'neu': 0.787, 'pos': 0.213, 'compound': 0.5574}
5	None	Question 2 as part of the "Be COVID Wise" series is here.How long does it take before one shows the symptoms of #Covid-1...	ScienceBagels	New Delhi, India	[(2, CARDINAL), (one, CARDINAL),]	{'neg': 0.0, 'neu': 0.871, 'pos': 0.129, 'compound': 0.4767}

Figure 12. Assigning data into the new column called sentiment

This shows how the sentiment looks for a tweet by assigning a value on the "compound" part of the sentiment column. If the compound value is negative then the overall sentiment is negative, or in the other case neutral if it is 0 and the overall sentiment is positive if the compound value is positive.

This is very useful, for instance when a new product is launched, we can gather data about that product and how the users are reacting to the market sentiment and how are they judging the new product.

As I mentioned before, there is a problem in the location column of the tweets as there are multiple locations for different areas of a country, such as London, London England, Dartmouth Canada, Canada, etc. To fix this issue I am going to make use of the GoogleMaps API, more specifically the Geolocation API as shown in the figure below.

First, I connected the program with the GoogleMaps API key and then I created a function to receive the full address for a tweet in JSON format. The next step is to extract only the last part of the address, which would be the country for all the tweets that have the location enabled.

```
import googlemaps #imports googlemaps package in order to make use of the google maps API
```

```
gmaps = googlemaps.Client(key = 'AIzaSyAtGQ9PjUEpNSglL3Bv2VZj9yVLOpX0aA8') #applying the API key
```

```
geocode_result = gmaps.geocode(tweet_df['location'][0]) #sample running on one of the

print(geocode_result) #gets the result from the geolocation API and prints it as a JSON format

print(geocode_result[0]['formatted_address']) #gets the formatted address as NEW YORK, NY, USA

print(geocode_result[0]['formatted_address'].split(",")[-1].strip()) #strips the last part of the address ("USA")
```


```
[{'address_components': [{'long_name': 'Dartmouth', 'short_name': 'Dartmouth', 'types': ['locality', 'political']}, {'long_nam
e': 'Halifax', 'short_name': 'Halifax', 'types': ['administrative_area_level_3', 'political']}, {'long_name': 'Halifax Regional
Municipality', 'short_name': 'Halifax Regional Municipality', 'types': ['administrative_area_level_2', 'political']}, {'long_na
me': 'Nova Scotia', 'short_name': 'NS', 'types': ['administrative_area_level_1', 'political']}, {'long_name': 'Canada', 'short_
name': 'CA', 'types': ['country', 'political']}], 'formatted_address': 'Dartmouth, NS, Canada', 'geometry': {'bounds': {'northe
ast': {'lat': 44.7412015, 'lng': -63.4955372}, 'southwest': {'lat': 44.6278566, 'lng': -63.6405576}}, 'location': {'lat': 44.66
60885, 'lng': -63.56756309999999}, 'location_type': 'APPROXIMATE', 'viewport': {'northeast': {'lat': 44.7412015, 'lng': -63.495
5372}, 'southwest': {'lat': 44.6278566, 'lng': -63.6405576}}}, 'place_id': 'ChIJszQD8t0jWksR5zFXGb4HUNo', 'types': ['locality',
'political']}]
```


```
Dartmouth, NS, Canada
Canada
```

Figure 13. Making use of the GoogleMaps API

Then, I created a Lambda function to apply the step above in all the tweets that have the location enabled and then creating a new column called "country" and assigning the values to each tweet.

```
def get_country(input): #convert the above into a functon to use as a lambda function for all the data
    try:
        output=gmaps.geocode(input)[0]['formatted_address'].split(",")[-1].strip()
    except:
        output="Error" #handles errors in data that there are special characters in the location column
    return output
```

```
tweet_df['country']=tweet_df['location'].apply(lambda x: "" if (not x.strip()) else get_country(x))
#creates another column on the dataframe and then applying the lambda function in order to get the location
```

```
tweet_df['country'].value_counts() #returns the countries with their values accordingly
```

```
                         2890
USA                      1760
UK                       1512
India                     940
Error                     436
                         ...
Al Jubail Saudi Arabia      1
Lake Ontario                1
Adirondack Mountains        1
Shopian 192303              1
Tanzania                    1
Name: country, Length: 185, dtype: int64
```

Figure 14. Assigning the values to the country column

As we can observe, there is 1760 tweets from the USA, 1512 from the UK, but also there are 436 cases the function cannot work as there might be special characters or invalid locations.

The following is an overview of the final dataset including the location which I extracted from the GoogleMaps API.

	geo	text	user	location	entities	sentiment	country
9990	None	#coronavirus related deaths 41% higher!Staggering deception from #Johnson and his #Tory governmentThe whole lot of th...	JosephR1201	Blantyre, Scotland UK	[(41%, PERCENT), (Johnson, PERSON)]	{'neg': 0.175, 'neu': 0.825, 'pos': 0.0, 'compound': -0.4926}	UK
9991	None	Using masks to fight #coronavirus would've been removed by Facebook last month.Not using masks to fight #cor...	MijanovicBlanka		[(Facebook, ORG), (last month, DATE)]	{'neg': 0.257, 'neu': 0.743, 'pos': 0.0, 'compound': -0.6369}	
9992	None	-Can't go to church.-Can't go fishing *by yourself* in your motor boat.-And now you can't buy seeds to garden at your hou...	PhilipImprescia		[]	{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}	
9993	None	No comments 🤣🤣🤣 #coronavirus #COVID—19 #coronavirusinindia	salilurunkar	पुणे	[(🤣, CARDINAL), (🤣🤣 #coronavirus, MONEY),]	{'neg': 0.306, 'neu': 0.694, 'pos': 0.0, 'compound': -0.296}	India
9994	None	Very proud to introduce EU#COVID19 Data Portal!Researchers worldwide can now access & submit #coronavirus data at unrival...	hideo84343927	東京 渋谷区	[(access &amp, ORG)]	{'neg': 0.0, 'neu': 0.813, 'pos': 0.187, 'compound': 0.5697}	Japan
9995	None	If your GB business operations in #Africa need support during the #coronavirus outbreak then can help. Please e...	AvrilBellon		[(Africa, LOC)]	{'neg': 0.0, 'neu': 0.661, 'pos': 0.339, 'compound': 0.7717}	
9996	None	CM Sindh President PPP Sindh Nisar A Khuhro & visited Larkana city & got briefing from ad...	AzadKha08835470		[(PPP Sindh, PERSON), (A Khuhro &amp, ORG), (Larkana city &amp, ORG)]	{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}	
9997	None	AP Govt's short-sighted attitude on #coronavirus and madness on publicity has infected11 Govt officials!!We demand a...	Venkate95668631	Bhimavaram, India	[(AP Govt's, PERSON), (Govt, ORG)]	{'neg': 0.356, 'neu': 0.644, 'pos': 0.0, 'compound': -0.7339}	India

Figure 15. Overview of the twitter data

Moreover, we can further clean the data by:

- Removing punctuation, special characters and numbers

```
tweet_df['text'] = tweet_df['text'].str.replace("[^a-zA-Z#]", " ")
```

- Removing the short words, for instance hmm, or, oh that do not have much use

```
tweet_df['text'] = tweet_df['text'].apply(lambda x: ' '.join([w for w in x.split() if len(w)>3]))
```

- Tokenization which is the process of splitting the words from each other

```
tokenized_tweet = tweet_df['text'].apply(lambda x: x.split())
```

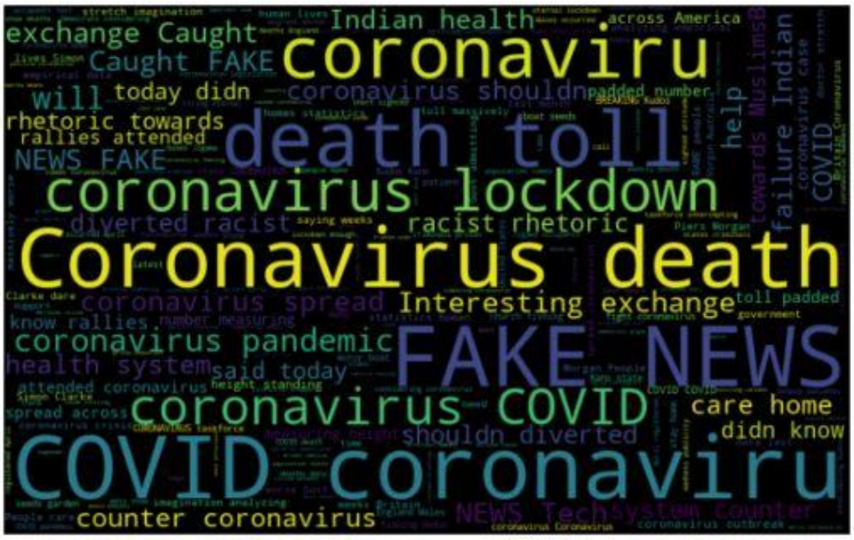
The following is the final dataset extracted from twitter:

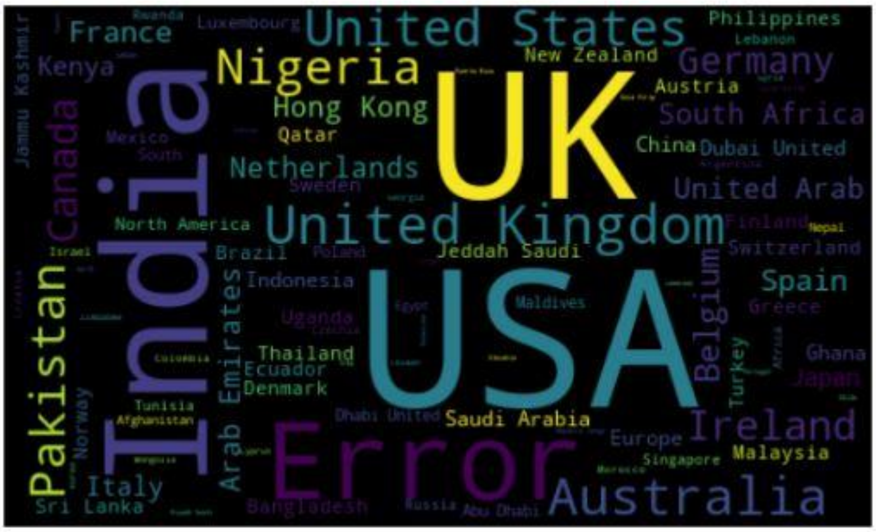
1	None	says global hunger could double #coronavirus blow says prepare living amid pande	NgoiePado		[(UN, ORG), (WHO, ORG)]	{'neg': 0.083, 'neu': 0.917, 'pos': 0.0, 'compound': -0.25}	
2	None	While Coronavirus Lockdown opens about plans meeting Bigg Boss bestie Read	asimcupcakes		[(Coronavirus Lockdown, PERSON), (Bigg Boss, FAC), (13, CARDINAL)]	{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}	
3	None	#selfisolationhelp #SelfIsolation #Isolation #coronavirus Play cards Patience	HeatherJShort		[, (36, CARDINAL), ,]	{'neg': 0.0, 'neu': 0.848, 'pos': 0.152, 'compound': 0.4003}	
4	None	Wanting know more about #coronavirus impact #genderequality have prepared special webpage that	alberto_soccol		[]	{'neg': 0.0, 'neu': 0.787, 'pos': 0.213, 'compound': 0.5574}	
5	None	Question part COVID Wise series here long does take before shows symptoms #Covid	ScienceBagels	New Delhi, India	[(2, CARDINAL), (one, CARDINAL),]	{'neg': 0.0, 'neu': 0.871, 'pos': 0.129, 'compound': 0.4767}	India
6	None	Standing head shoulders above most people Wendell Quinn height only surpassed amount caring	PlanetKinsman	Home o/t Range in the CA sun	[(Wendell Quinn's, ORG)]	{'neg': 0.0, 'neu': 0.856, 'pos': 0.144, 'compound': 0.4939}	Error
7	None	failure #Indian health system counter #coronavirus shouldn diverted racist rhetoric towards MuslimsB	NumaniAkbar	India	[, (Indian, NORP)]	{'neg': 0.313, 'neu': 0.687, 'pos': 0.0, 'compound': -0.8074}	India
8	None	This nation over confidence know have challenged #coronavirus every occasion keep sayin	Hania_Mir22	Syria	[]	{'neg': 0.062, 'neu': 0.793, 'pos': 0.145, 'compound': 0.4404}	Syria
9	None	#Coronavirus updates futures plunged below first time history Cases rise more than million	NgoiePado		[(US, GPE), (0, CARDINAL), (first, ORDINAL), (history•, GPE), (Cases, ORG), (more than 2.4 million, MONEY)]	{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}	

From this data we could also generate a wordcloud to see what people are writing regarding the coronavirus hashtag and what locations they are mostly based.

```
all_words = ' '.join([text for text in tweet_df['text']])
from wordcloud import WordCloud
wordcloud = WordCloud(width=800, height=500, random_state=21, max_font_size=110).generate(all_words)

plt.figure(figsize=(10, 7))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
plt.show()
```





2) Predicting the virus outbreak according to the coronavirus dataset available.

For this part of the project I decided to gather several datasets regarding the Coronavirus, which are available on Github repository. These datasets include the confirmed cases worldwide, the confirmed deaths and the confirmed number of recoveries, all from the 22nd of January until the 20th of April.

EXTRACTING THE DATA FROM GITHUB

First of all, I loaded all the datasets separately into a pandas dataframe as seen below:

Total number of confirmed cases:

```
corona_confirmed_df = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covi
```

```
corona_confirmed_df.head()
```

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	...	4/10/20	4/11/20	4/12/20	4/13/20	4/14/20	4/15/20
0	NaN	Afghanistan	33.0000	65.0000	0	0	0	0	0	0	...	521	555	607	665	714	784
1	NaN	Albania	41.1533	20.1683	0	0	0	0	0	0	...	416	433	446	467	475	494
2	NaN	Algeria	28.0339	1.6596	0	0	0	0	0	0	...	1761	1825	1914	1983	2070	2160
3	NaN	Andorra	42.5063	1.5218	0	0	0	0	0	0	...	601	601	638	646	659	673
4	NaN	Angola	-11.2027	17.8739	0	0	0	0	0	0	...	19	19	19	19	19	19

5 rows × 93 columns

Figure 16. Confirmed cases by country and by date

Total number of deaths:

```
corona_death_df = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_
```

```
corona_death_df.head()
```

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	...	4/10/20	4/11/20	4/12/20	4/13/20	4/14/20	4/15/20
0	NaN	Afghanistan	33.0000	65.0000	0	0	0	0	0	0	...	15	18	18	21	23	25
1	NaN	Albania	41.1533	20.1683	0	0	0	0	0	0	...	23	23	23	23	24	25
2	NaN	Algeria	28.0339	1.6596	0	0	0	0	0	0	...	256	275	293	313	326	336
3	NaN	Andorra	42.5063	1.5218	0	0	0	0	0	0	...	26	26	29	29	31	33
4	NaN	Angola	-11.2027	17.8739	0	0	0	0	0	0	...	2	2	2	2	2	2

5 rows × 93 columns

Figure 17. Confirmed deaths by country and by date

Total number of recoveries:

```
corona_recovered_df = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covi
```

```
corona_recovered_df.head()
```

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	...	4/10/20	4/11/20	4/12/20	4/13/20	4/14/20	4/15/20
0	NaN	Afghanistan	33.0000	65.0000	0	0	0	0	0	0	...	32	32	32	32	40	43
1	NaN	Albania	41.1533	20.1683	0	0	0	0	0	0	...	182	197	217	232	248	251
2	NaN	Algeria	28.0339	1.6596	0	0	0	0	0	0	...	405	460	591	601	691	708
3	NaN	Andorra	42.5063	1.5218	0	0	0	0	0	0	...	71	71	128	128	128	169
4	NaN	Angola	-11.2027	17.8739	0	0	0	0	0	0	...	2	4	4	4	5	5

5 rows × 93 columns

Figure 18. Confirmed number of recoveries by country and by date

DATA PREPROCESSING

The next step is to process the data and merging all the datasets into one. I took an approach of converting this timeseries data by flattening it out, rather than having one row for each Province/State.

DATE → Rows for each date

For this conversion, I used the "melt" function of the dataframe which will take all the keys which we want to keep and convert the remaining columns into rows.

```
corona_death_df = corona_death_df.melt(id_vars=['Province/State', 'Country/Region','Lat','Long'])
corona_death_df = corona_death_df.rename({'variable':'Date','value':'Death'}, axis='columns')
corona_death_df.head()
```

	Province/State	Country/Region	Lat	Long	Date	Death
0	NaN	Afghanistan	33.0000	65.0000	1/22/20	0
1	NaN	Albania	41.1533	20.1683	1/22/20	0
2	NaN	Algeria	28.0339	1.6596	1/22/20	0
3	NaN	Andorra	42.5063	1.5218	1/22/20	0
4	NaN	Angola	-11.2027	17.8739	1/22/20	0

Figure 19. Performing conversion using Melt function

The function above will be used for the two remaining datasets in the same way.

The following step is to combine all the 3 dataframes into one, by using the "join" function in Python, as shown below:

```
combined_df = [corona_confirmed_df, corona_death_df, corona_recovered_df]
combined_df = [df.set_index(['Province/State','Country/Region', 'Lat', 'Long', 'Date']) for df in combined_df]
combined_df = combined_df[0].join(combined_df[1:])
```

```
combined_df = combined_df.reset_index()
```

```
combined_df.head()
```

					Confirmed	Death	Recovered
Province/State	Country/Region	Lat	Long	Date			
NaN	Afghanistan	33.0000	65.0000	1/22/20	0.0	0.0	0.0
	Albania	41.1533	20.1683	1/22/20	0.0	0.0	0.0
	Algeria	28.0339	1.6596	1/22/20	0.0	0.0	0.0
	Andorra	42.5063	1.5218	1/22/20	0.0	0.0	0.0
	Angola	-11.2027	17.8739	1/22/20	0.0	0.0	0.0

Figure 20. Joining the Dataframes in one

If we observe the values of the Lat/Lon, Confirmed, Death and Recovered are not numbers so I had to convert their values into numbers, to make it easier to work with them afterwards.

```
combined_df[['Lat', 'Long', 'Confirmed', 'Death', 'Recovered']] = combined_df[['Lat', 'Long', 'Confirmed', 'Death', 'Recovered']
```

```
combined_df[['Date']] = combined_df[['Date']].apply(pd.to_datetime)
```

```
combined_df.dtypes
```

```
Province/State            object
Country/Region            object
Lat                      float64
Long                     float64
Date              datetime64[ns]
Confirmed                float64
Death                    float64
Recovered                float64
dtype: object
```

```
combined_df.head()
```

	Province/State	Country/Region	Lat	Long	Date	Confirmed	Death	Recovered
0	NaN	Afghanistan	33.0000	65.0000	2020-01-22	0.0	0.0	0.0
1	NaN	Albania	41.1533	20.1683	2020-01-22	0.0	0.0	0.0
2	NaN	Algeria	28.0339	1.6596	2020-01-22	0.0	0.0	0.0
3	NaN	Andorra	42.5063	1.5218	2020-01-22	0.0	0.0	0.0
4	NaN	Angola	-11.2027	17.8739	2020-01-22	0.0	0.0	0.0

Figure 21. Converting the values into numbers

The following step would be to check all the countries that have null values in the "State" column as we can observe from above there are many null values.

```
combined_df[combined_df.State.isnull()]['Country'].value_counts()
```

```
Angola              89
Gambia              89
Congo (Kinshasa)    89
Zimbabwe            89
Bangladesh          89
                    ..
Mozambique          89
Mongolia            89
Tunisia             89
Mauritius           89
Colombia            89
Name: Country, Length: 182, dtype: int64
```

Figure 22. Null values for the States

In order to fill these values, I made use of the GoogleMaps API again. However, this time I had to do a reverse geocoding instead of the process done in the previous dataset. A reverse geocoding will take the Latitude and Longitude coordinates and then make requests to Google in order to receive the response. In this case the response would be the name of the States.

```
import googlemaps
```

```
gmaps = googlemaps.Client(key = 'AIzaSyAtGQ9PjUEpNSglL3Bv2VZj9yVLOpX0aA8') #applying the API key

geocode_result = gmaps.reverse_geocode((-25.09306, -57.52361))

print(geocode_result[0]) #gets the result from the geolocation API and prints it as a JSON format

print(geocode_result[0]['formatted_address']) #gets the formatted address as NEW YORK, NY, USA

print(geocode_result[0]['formatted_address'].split(",")[-2].strip(" ")) #strips the last part of the address ("USA")
```


```
{'address_components': [{'long_name': 'Cerro Corá', 'short_name': 'Cerro Corá', 'types': ['route']}, {'long_name': 'Villa Haye
s', 'short_name': 'Villa Hayes', 'types': ['locality', 'political']}, {'long_name': 'Villa Hayes', 'short_name': 'Villa Hayes',
'types': ['administrative_area_level_2', 'political']}, {'long_name': 'Presidente Hayes', 'short_name': 'Presidente Hayes', 'ty
pes': ['administrative_area_level_1', 'political']}, {'long_name': 'Paraguay', 'short_name': 'PY', 'types': ['country', 'politi
cal']}], 'formatted_address': 'Cerro Corá, Villa Hayes, Paraguay', 'geometry': {'bounds': {'northeast': {'lat': -25.0926905, 'l
ng': -57.5229341000001}, 'southwest': {'lat': -25.0933372, 'lng': -57.5238388}}, 'location': {'lat': -25.0930139, 'lng': -57.5
233865}, 'location_type': 'GEOMETRIC_CENTER', 'viewport': {'northeast': {'lat': -25.0916648697085, 'lng': -57.5220374697085},
'southwest': {'lat': -25.0943628302915, 'lng': -57.52473543029151}}}, 'place_id': 'ChIJLTIHdZqYXZQRoGdPsuXIBPQ', 'types': ['rou
te']}
```


```
Cerro Corá, Villa Hayes, Paraguay
Villa Hayes
```

Figure 23. Implementing Reverse Geocoding via GoogleMap API

As shown above after performing the request, I got back as a response the location as a JSON format and then I had to do a "split" and "strip" function in order to get back the state. For instance, Cerro Cora, Villa Hayes, Paraguay, I had to take the middle value, which in this case is Villa Hayes.

Then, I created a Lambda function to perform this in all the dataset and fill the null values for each country. However, we are going to see later that not all values are going to be fixed, hence another approach will be taken later.

```python
import re

english_check = re.compile(r'[a-z]')

def get_state(lat, longi):
    try:
        output = gmaps.reverse_geocode((lat, longi))[0]['formatted_address'].split(",")[-2].strip()
        if(len(output.split(" ")) > 1):
            output = "TBF"

        if not english_check.match(output.lower()):
            output = "TBF"

    except:
        print("Error in Reverse Geocoding")
        output = "TBF"

    return output
```

```python
combined_df["state_cleaned"]=combined_df[combined_df.State.isnull()][['Lat','Long']].apply(lambda x : get_state(x['Lat'], x['Lon
```

Figure 24. Creating the function to use in the dataset

The function above is going to try and impute as many values as possible by using the Lat and Long for each state and then add the values into a new column called “state_cleaned”.

Then, my next step was to create a copy of the dataframe, just in case mistakes are made further during processing and then we can easily load the previous dataframe.

```python
combined_df[combined_df.state_cleaned.notna()]
```

	State	Country	Lat	Long	Date	Confirmed	Death	Recovered	state_cleaned
0	NaN	Afghanistan	33.000000	65.000000	2020-01-22	0.0	0.0	0.0	Baghran
1	NaN	Albania	41.153300	20.168300	2020-01-22	0.0	0.0	0.0	E852
2	NaN	Algeria	28.033900	1.659600	2020-01-22	0.0	0.0	0.0	Timokten
3	NaN	Andorra	42.506300	1.521800	2020-01-22	0.0	0.0	0.0	TBF
4	NaN	Angola	-11.202700	17.873900	2020-01-22	0.0	0.0	0.0	Sautari
...	...	...	...	...	...	...	...	...	...
23489	NaN	Malawi	-13.254308	34.301525	2020-04-19	17.0	2.0	3.0	Nkhotakota
23492	NaN	South Sudan	6.877000	31.307000	2020-04-19	4.0	0.0	0.0	Akobo
23493	NaN	Western Sahara	24.215500	-12.885800	2020-04-19	6.0	0.0	0.0	TBF
23494	NaN	Sao Tome and Principe	0.186360	6.613081	2020-04-19	4.0	0.0	0.0	st
23495	NaN	Yemen	15.552727	48.516388	2020-04-19	1.0	0.0	0.0	TBF

Figure 25. Copy of the dataframe along with the new column on the right

As we can observe, many values have been filled, however there are too many other values that contain “TBF”, meaning that I had to take another approach in order to fill them. If we apply the following code, we can assign from the “state” column values into the “state_cleaned” column.

```
combined_int_df[combined_int_df.State.notna()]["State"].count()
```

7298

```
import numpy
combined_int_df["state_cleaned"] = combined_int_df.apply(lambda x: x['State'] if (numpy.all(pd.notnull(x['State']))) else x['sta
```

```
combined_int_df[combined_int_df.State.notna()]
```

	State	Country	Lat	Long	Date	Confirmed	Death	Recovered	state_cleaned
8	Australian Capital Territory	Australia	-35.4735	149.0124	2020-01-22	0.0	0.0	0.0	Australian Capital Territory
9	New South Wales	Australia	-33.8688	151.2093	2020-01-22	0.0	0.0	0.0	New South Wales
10	Northern Territory	Australia	-12.4634	130.8456	2020-01-22	0.0	0.0	0.0	Northern Territory
11	Queensland	Australia	-28.0167	153.4000	2020-01-22	0.0	0.0	0.0	Queensland
12	South Australia	Australia	-34.9285	138.6007	2020-01-22	0.0	0.0	0.0	South Australia
...	...	...	...	...	...	...	...	...	...
23482	British Virgin Islands	United Kingdom	18.4207	-64.6400	2020-04-19	4.0	1.0	2.0	British Virgin Islands
23483	Turks and Caicos Islands	United Kingdom	21.6940	-71.7979	2020-04-19	11.0	1.0	0.0	Turks and Caicos Islands
23488	Bonaire, Sint Eustatius and Saba	Netherlands	12.1784	-68.2385	2020-04-19	5.0	0.0	0.0	Bonaire, Sint Eustatius and Saba
23490	Falkland Islands (Malvinas)	United Kingdom	-51.7963	-59.5236	2020-04-19	11.0	0.0	3.0	Falkland Islands (Malvinas)
23491	Saint Pierre and Miquelon	France	46.8852	-56.3159	2020-04-19	1.0	0.0	0.0	Saint Pierre and Miquelon

7298 rows × 9 columns

Figure 26. Assigning values from State to state_cleaned

The following are the Countries that need to be filled with a value under the “state_cleaned” column.

```
combined_int_df[combined_df.state_cleaned=="TBF"]
```

	State	Country	Lat	Long	Date	Confirmed	Death	Recovered	state_cleaned
3	NaN	Andorra	42.506300	1.521800	2020-01-22	0.0	0.0	0.0	TBF
5	NaN	Antigua and Barbuda	17.060800	-61.796400	2020-01-22	0.0	0.0	0.0	TBF
6	NaN	Argentina	-38.416100	-63.616700	2020-01-22	0.0	0.0	0.0	TBF
7	NaN	Armenia	40.069100	45.038200	2020-01-22	0.0	0.0	0.0	TBF
16	NaN	Austria	47.516200	14.550100	2020-01-22	0.0	0.0	0.0	TBF
...	...	...	...	...	...	...	...	...	...
23476	NaN	Saint Kitts and Nevis	17.357822	-62.782998	2020-04-19	14.0	0.0	0.0	TBF
23479	NaN	Kosovo	42.602636	20.902977	2020-04-19	510.0	12.0	93.0	TBF
23484	NaN	MS Zaandam	0.000000	0.000000	2020-04-19	9.0	2.0	0.0	TBF
23493	NaN	Western Sahara	24.215500	-12.885800	2020-04-19	6.0	0.0	0.0	TBF
23495	NaN	Yemen	15.552727	48.516388	2020-04-19	1.0	0.0	0.0	TBF

9523 rows × 9 columns

Figure 27. List of countries that need to be filled with the state_cleaned value

To fill the values with their corresponding states I made use of another dataset, available on Github that contains values for Countries and their respective Capitals.

```
capital_df = pd.read_csv('https://raw.githubusercontent.com/icyrockcom/country-capitals/master/data/country-list.csv')
```

```
capital_df.head()
```

	country	capital	type
0	Abkhazia	Sukhumi	countryCapital
1	Afghanistan	Kabul	countryCapital
2	Akrotiri and Dhekelia	Episkopi Cantonment	countryCapital
3	Albania	Tirana	countryCapital
4	Algeria	Algiers	countryCapital

```
capital_df['country'].replace({"United Kingdom; England": "United Kingdom"}, inplace=True)
```

```
capital_df['country'] = capital_df.country.str.lower()
```

```
capital_df.head()
```

	country	capital	type
0	abkhazia	Sukhumi	countryCapital
1	afghanistan	Kabul	countryCapital
2	akrotiri and dhekelia	Episkopi Cantonment	countryCapital
3	albania	Tirana	countryCapital
4	algeria	Algiers	countryCapital

Figure 28. Countries and their capitals

Looking at the picture above, I loaded the dataset into a dataframe called "capital_df" and then I replaced United Kingdom, England with United Kingdom only, as this is how it appears on the dataset we are using.

The next step is to convert all the data from the Capitals dataset into lowercase letters in order to perform the comparison with the other dataset.

```
def lookup_capital(country):
    try:
        capital=capital_df.loc[country.lower()][0]
    except:
        capital="TBF"
    return capital
```

```
combined_int_df.apply(lambda x: lookup_capital(x['Country']) if (x['state_cleaned'] == "TBF") else x['state_cleaned'], axis=1)
```

```
0                      Baghran
1                         E852
2                     Timokten
3             Andorra la Vella
4                      Sautari
                  ...
23491   Saint Pierre and Miquelon
23492                    Akobo
23493                  El Aaiún
23494                       st
23495                    Sanaá
Length: 23496, dtype: object
```

```
combined_int_df['state_cleaned'] = combined_int_df.apply(lambda x: lookup_capital(x['Country']) if (x['state_cleaned']=="TBF") e
```

Figure 29. Function to lookup the capitals of respective countries

As shown in Figure 29 above, I created a function called "lookup_capital" and compared it with the lowercase countries in the main dataset. The next step was to apply all the corresponding values in the main dataset column, "state_cleaned".

This is how the final dataset looks like after pre-processing it:

	Country	Lat	Long	Date	Confirmed	Death	Recovered	state_cleaned
	Afghanistan	33.000000	65.000000	2020-01-22	0.0	0.0	0.0	Baghran
	Albania	41.153300	20.168300	2020-01-22	0.0	0.0	0.0	Tirana
	Algeria	28.033900	1.659600	2020-01-22	0.0	0.0	0.0	Timokten
	Andorra	42.506300	1.521800	2020-01-22	0.0	0.0	0.0	Andorra la Vella
	Angola	-11.202700	17.873900	2020-01-22	0.0	0.0	0.0	Sautari
	...	...	...	...	...	...	...	...
	France	46.885200	-56.315900	2020-04-19	1.0	0.0	0.0	Saint Pierre and Miquelon
	South Sudan	6.877000	31.307000	2020-04-19	4.0	0.0	0.0	Akobo
	Western Sahara	24.215500	-12.885800	2020-04-19	6.0	0.0	0.0	El Aaiún
	Sao Tome and Principe	0.186360	6.613081	2020-04-19	4.0	0.0	0.0	st
	Yemen	15.552727	48.516388	2020-04-19	1.0	0.0	0.0	Sanaá

Figure 30. Final Dataset

DESCRIPTION OF THE MACHINE LEARNING PIPELINE (Methodology Adopted)

I decided to use Support Vector Machine (SVM) for the purpose of this challenge. The reason is that because of the gradual use of SVM in different data science problems, it has become a powerful tool that is widely used in clustering, regression or classification problems. The challenge is to try and predict the number of coronavirus cases given the dataset made available.

Firstly, I got the total number of cases for the whole period in all countries, and the total number of recovered cases from the following code:

```
dates = confirmed.keys()
world_cases = []
total_deaths = []
mortality_rate = []
total_recovered = []

for i in dates:
    confirmed_sum = confirmed[i].sum()
    death_sum = deaths[i].sum()
    recovered_sum = recovered[i].sum()
    world_cases.append(confirmed_sum)
    total_deaths.append(death_sum)
    mortality_rate.append(death_sum/confirmed_sum)
    total_recovered.append(recovered_sum)
```

```
days_since_1_22 = np.array([i for i in range(len(dates))]).reshape(-1,1)
world_cases = np.array(world_cases).reshape(-1,1)
total_deaths = np.array(total_deaths).reshape(-1,1)
total_recovered = np.array(total_recovered).reshape(-1,1)
```

```
confirmed_sum
```

```
2401378
```

```
recovered_sum
```

```
623903
```

Figure 31. Getting the total sum of cases and total sum of recovered cases

Then the next step was creating the array of future forecasting, for the next 10 days as following.

```
#future forecasting for the next 10 days

days_in_future = 10
future_forecast = np.array([i for i in range(len(dates)+days_in_future)]).reshape(-1,1)
adjusted_dates = future_forecast[:-10]
future_forecast
```

Furthermore, I applied the following function to the dataset to extract the total number of confirmed coronavirus cases per each country.

```
country_corona_confirmed_df = []
no_cases = []

for i in unique_countries:
    cases = latest_confirmed[corona_confirmed_df['Country/Region']==i].sum()
    if cases > 0:
        country_corona_confirmed_df.append(cases)
    else:
        no_cases.append(i)

for i in no_cases:
    unique_countries.remove(i)

unique_countries = [k for k, v in sorted(zip(unique_countries, country_corona_confirmed_df), key=operator.itemgetter(1))]
for i in range(len(unique_countries)):
    country_corona_confirmed_df[i] = latest_confirmed[corona_confirmed_df['Country/Region']==unique_countries[i]].sum()
```

```
print('CONFIRMED CASES PER COUNTRY:')
for i in range(len(unique_countries)):
    print(f'{unique_countries[i]}: {country_corona_confirmed_df[i]} cases')
```

```
Hungary: 1916 cases
Greece: 2235 cases
Bangladesh: 2456 cases
Moldova: 2472 cases
Algeria: 2629 cases
Thailand: 2765 cases
Argentina: 2839 cases
Morocco: 2855 cases
Egypt: 3144 cases
South Africa: 3158 cases
Luxembourg: 3550 cases
Finland: 3783 cases
```

Figure 32. Total number of cases per each country

The next step was to build the SVM model based on the dataset as following:

```
#Building the SVM model:

kernel = ['poly', 'sigmoid', 'rbf']
c = [0.01, 0.1, 1, 10]
gamma = [0.01, 0.1, 1]
epsilon = [0.01, 0.1, 1]
shrinking = [True, False]
svm_grid = {'kernel': kernel, 'C': c, 'gamma': gamma, 'epsilon': epsilon, 'shrinking': shrinking}
svm = SVR()
X_train_confirmed, X_test_confirmed, y_train_confirmed, y_test_confirmed = train_test_split(days_since_1_22, world_cases, test_s

svm_search = RandomizedSearchCV(svm, svm_grid, scoring='neg_mean_squared_error', cv=3, return_train_score=True, n_jobs=-1, n_ite
svm_search.fit(X_train_confirmed, y_train_confirmed)
```

Figure 33. Building the SVM model

The model then would find the best parameters as shown on the figure below:

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done 240 out of 240 | elapsed:  1.9min finished
C:\Users\user\Anaconda3\lib\site-packages\sklearn\utils\validation.py:724: DataConversionWarning: A column-vector y was passed
when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)

RandomizedSearchCV(cv=3, error_score='raise-deprecating',
                   estimator=SVR(C=1.0, cache_size=200, coef0=0.0, degree=3,
                                 epsilon=0.1, gamma='auto_deprecated',
                                 kernel='rbf', max_iter=-1, shrinking=True,
                                 tol=0.001, verbose=False),
                   iid='warn', n_iter=80, n_jobs=-1,
                   param_distributions={'C': [0.01, 0.1, 1, 10],
                                        'epsilon': [0.01, 0.1, 1],
                                        'gamma': [0.01, 0.1, 1],
                                        'kernel': ['poly', 'sigmoid', 'rbf'],
                                        'shrinking': [True, False]},
                   pre_dispatch='2*n_jobs', random_state=None, refit=True,
                   return_train_score=True, scoring='neg_mean_squared_error',
                   verbose=1)
```

```
svm_search.best_params_
```

```
{'shrinking': False, 'kernel': 'poly', 'gamma': 1, 'epsilon': 0.1, 'C': 1}
```

Figure 34. Best parameters for the model

The following graph shows the total number of coronavirus cases over time since starting to gather the data.

```
plt.figure(figsize=(10, 8))
plt.plot(adjusted_dates, world_cases)
plt.title('Number of coronavirus cases over time', size=20)
plt.xlabel('Days since 22nd of January', size=20)
plt.ylabel('Number of cases', size=20)
plt.xticks(size=15)
plt.yticks(size=15)
plt.show()
```

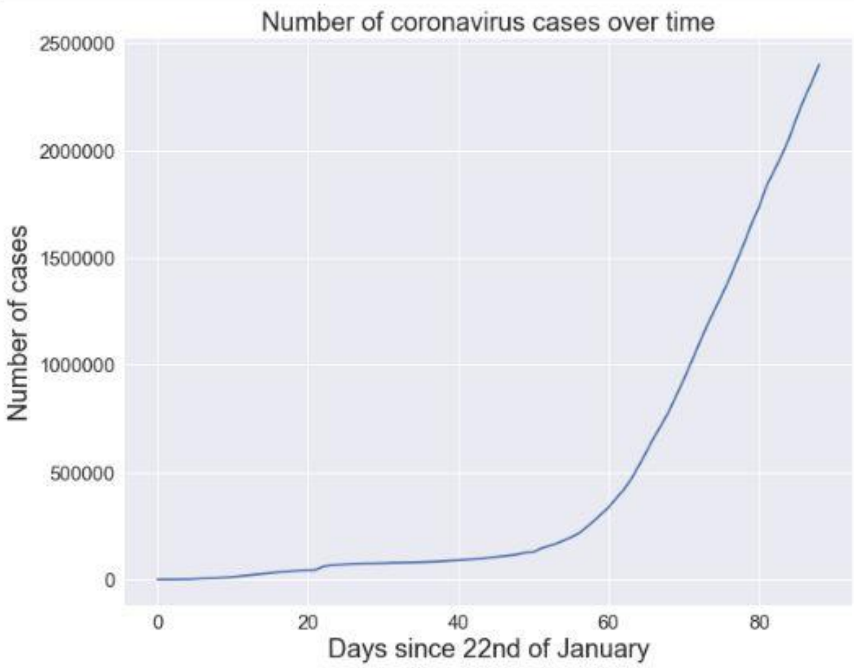


Figure 35. Total number of cases over time

Now the following will show the total number of cases over time vs the SVM predictions:

```
plt.plot(future_forecast, svm_pred, linestyle='dashed', color='red')
plt.title('Number of coronavirus cases over time', size=20)
plt.xlabel('Days since 22nd of January', size=20)
plt.ylabel('Number of cases', size=20)
plt.legend(['Confirmed Cases', 'SVM Predictions'])
plt.xticks(size=15)
plt.yticks(size=15)
plt.show()
```

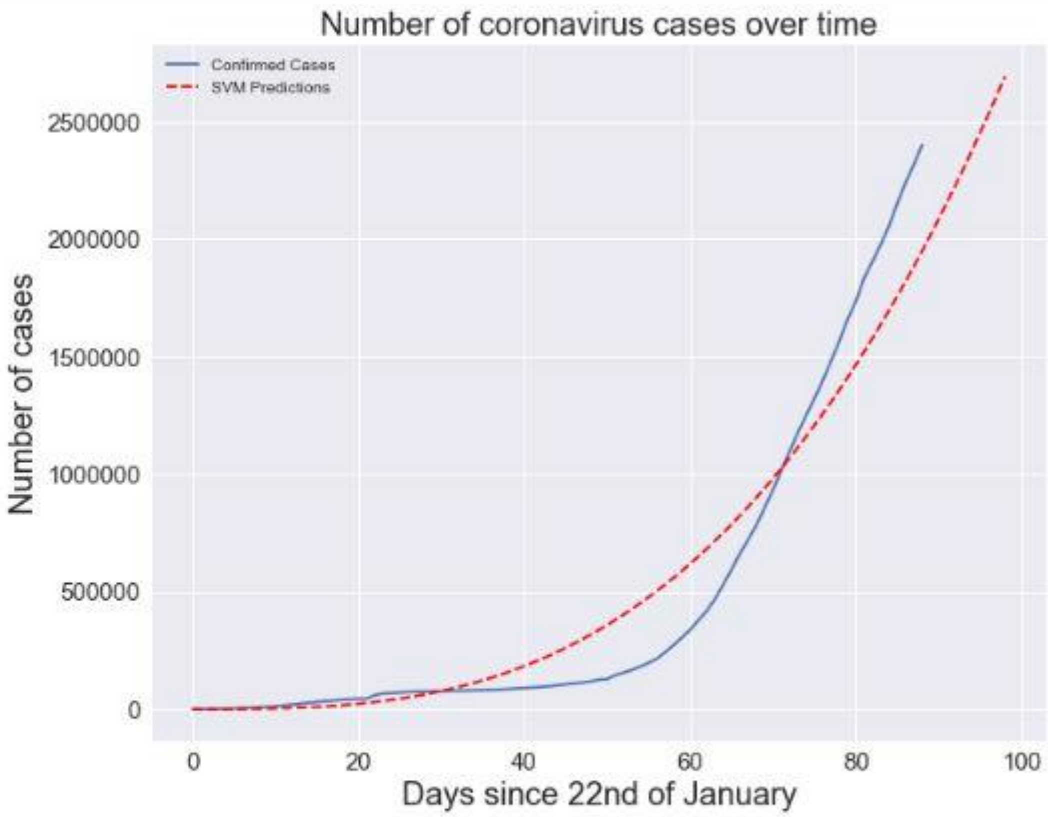


Figure 36. Cases vs SVM model

By running the following we can also get predictions for the next 10 days of the total number of coronavirus cases.

```
#Predictions for the next 10 days using SVM

print('SVM FUTURE PREDICTIONS: ')
set(zip(future_forecast_dates[-10:], svm_pred[-10:]))
```

```
SVM FUTURE PREDICTIONS:

{('04/20/2020', 2017494.498916527),
 ('04/21/2020', 2086247.9769438708),
 ('04/22/2020', 2156546.4124907455),
 ('04/23/2020', 2228406.972061058),
 ('04/24/2020', 2301846.821670433),
 ('04/25/2020', 2376883.128311058),
 ('04/26/2020', 2453533.056045433),
 ('04/27/2020', 2531813.7738188705),
 ('04/28/2020', 2611742.4456938705),
 ('04/29/2020', 2693336.2386626205)}
```

Figure 37. Predictions for the next 10 days

LITERATURE REVIEW OF RELEVANT MACHINE LEARNING TECHNIQUES

Data science is part of machine learning, which itself is part of Artificial Intelligence. Machine learning, is widely used nowadays for different purposes for some reasons:

- Because nowadays the challenges we are facing are high-dimensional.
- Because we have more and more data from various data sources that help to build different models to solve these high-dimensional challenges.
- We can integrate these models in different software that are more in demand by the industry.

To solve these challenges, it requires a big number of variables that, influence the kind of different observations we make in business and science and humans are in many cases incapable of reading and making observations at 3 dimensions. Hence, machine learning is fundamentally important, lately the decision making has been shifted partially to the machine. ML can work and make observations in these high dimensional spaces and generate good solutions as well. There are many techniques used in machine learning depending on the challenge we face:

- Regression
- Classification
- Clustering
- Natural language processing
- Neural networks

Before moving to the techniques mentioned above there are 3 main types of machine learning problems:

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

In supervised learning, there is a problem that involves using the model to learn a mapping between the input variables which we already have and the target output variable. Applications in which the training data comprises examples of the input vectors along with their corresponding target vectors are known as supervised learning problems [1].

Models have to be fit on training data, composed of input values and output values and predictions have to be made on test data where only the input values are known and the output values from the model are compared to with the target variable withheld, these estimating the ability of the model.

In order to measure the hypothesis accuracy, we give it a test set of examples that are different from the training set [2].

Going back to the previous part, there are two main groups of supervised learning techniques:

There is ***regression***, that involves the prediction of a numerical value and there is ***classification***, that involves the prediction of a class label. For both techniques, there might be more than one input variable, either numerical or categorical.

For instance, a classification problem would be a model that can detect spam emails, or fraud detection. On the other hand, a regression problem might be predicting house prices in a specific area based on other variables or like in this case the prediction of coronavirus confirmed cases.

There are other algorithms that are described as "supervised" algorithms in machine learning. Some examples include: ***Decision Trees*** and ***Support Vector Machines***.

In this project I used Support Vector Machines (SVM), as the goal was to find a f(x) to the function f(x) that underlies the predictive relationship between the inputs and outputs [3]. Support Vector Machines within the area of structural risk minimization and statistical learning theory, have demonstrated to successfully work on numerous forecasting problems. SVMs have been widely used in regression estimation problems and many pattern recognitions challenges [8]. They have been applied to the challenges of forecasting, dependency estimation and the construction of intelligent machines [4].

<u>**Unsupervised learning**</u>, on the other hand describes challenges and problems that involve using a model to extract or describe relationships in the data itself.

There are many types of unsupervised learning algorithm, but some of the most widely used ones involve, ***clustering*** that entails finding groups in the data and ***density estimation*** that includes summarizing the data distribution.

There are additional unsupervised methods, such as ***visualization*** that involves visualizing, or grouping the data in plots or graphs and ***projection methods*** that include reducing the data dimensionality.

EVALUATION AGAINST APPROPRIATE BASELINE TECHNIQUES

There have not been too many models published on fighting the spread of coronavirus, however there has been a significant amount of research done on forecasting other outbreaks and seasonal flus [7].

For instance, there is a model that predicts weather as infected patient would survive or not, based on a XG-Boost model. This model takes into consideration the patient's age and other risk factors. This type of model is very useful, because gives insights on who should isolate from the virus the most.

Another model for the fight against the virus is mining the social media. However, the social media is very noisy, so it is quite difficult to come up with a very efficient model. There are other publications based on previous viruses, such as Influenza [6].

From the twitter data I presented on this assignment, we can observe that there is a large usage of the word "Fake", meaning that there is a lot of materials and news out there on social media twitter, that are not true.

Given the fact that this is relatively a new virus, there is yet to understand its nature and other conditions that might affect the spread, such as weather conditions, or previous health conditions in individuals that are infected, so the results we get from such machine learning algorithms might not be ideal and very accurate. However, there is another algorithm that might be more efficient, as it makes use of other datasets and conditions, such as getting the population for specific countries and the weather history of a given location.

This model uses a Kalman filter, which is an algorithm that takes measurements that are observed over time. These observations might contain statistical noise, and inaccuracies and produces estimates of unknown variables. These unknown variables tend to be more accurate using the Kalman model, than other models that are based on single measurements only.

CONCLUSION

Finally, and maybe most importantly, for making progress, I genuinely believe that different organizations and places should start sharing their data. However, it is essential that the data needs to be de-anonymized and personal identifiable information should be stripped off, and then shared. There is a vast number of data behind closed doors, such as hospitals and clinics.

Furthermore, patient confidentiality is very important, but many research groups use it only as an excuse to not share data. I believe that these steps are necessary for Artificial Intelligence to be enabled to address this health challenge our society is facing lately.

REFERENCES

1 – Bishop, C., 2006. "*Pattern Recognition and Machine Learning*". Page 3.

2 – Peter, N., 2015. "*Artificial Intelligence: A modern approach*". Page 395.

3 – Jerome H. Friedman, 2016. "*Elements of statistical learning: data mining, interference and prediction", 2nd edition.* Page 28.

4 – E. Sami, 2012. *"Support Vector Machines for classification and locating faults on transmission lines"*, vol. 12, pp. 1650–1658.

5 – Acke, E. and Cave, N.J., 2020. "*A serological survey of canine respiratory coronavirus in New Zealand. New Zealand Veterinary Journal",* 68(1), pp.54-59.

6 – Nsoesie, E.O., 2014. "*A systematic review of studies on forecasting the dynamics of influenza outbreaks. Influenza and other respiratory viruses"*, 8(3), pp.309-316.

7 – World Health Organization, 2020. "*Coronavirus disease 2019 (COVID19)": situation report*, 67.

8 – Kim,H.Y. (2005). *"An application of support vector machines in bankruptcy prediction model, Expect Systems with Application"*.

9 – D. Fletcher and E. Goss, *"Forecasting with neural networks— an application using bankruptcy data*", Inf vol. 24, (1993), pp. 159–167.