Configuration Manual

Using Natural Language Processing Techniques to Analyze the Impact of Covid-19 on Stock Market

MSc Research Project
Data Analytics

Wei He
Student ID: x18144489

School of Computing
National College of Ireland

Supervisor: Dr. Majid Latifi

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Wei He
Student ID:	X18144489
Programme:	Master of Science in Data Analytics **Year:** 2021
Module:	Research Project
Lecturer:	Dr. Majid Latifi
Submission Due Date:	16 Aug 2021
Project Title:	Using Natural Language Processing Techniques to Analyze the Impact of Covid-19 on Stock Market
Word Count:	3459 **Page Count:** 89

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Wei He
Date:	16 Aug 2021

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

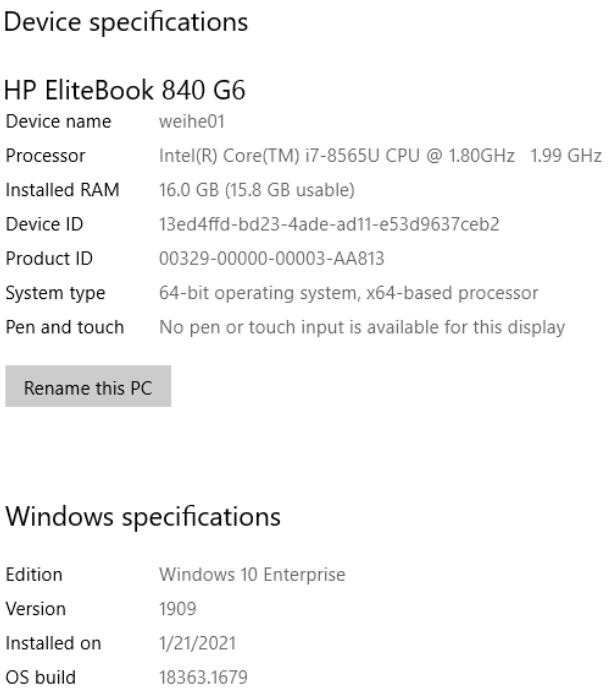
Wei He
Student ID: x18144489

1 Introduction

This configuration manual describes the hardware and software setup and the software installation steps for this project. The steps are used for the project Using Natural Language Processing Techniques to Analyze the Impact of Covid-19 on Stock Market. I covered the below sections: data collect, pre-processsing, implementation and evaluation.

2 Hardware Configuration

This project tasks were run under this HP EliteBook 840 G6 laptop. It has 64bit Windows 10 OS. Please see the detailed specification in Figure 1



Device specifications

HP EliteBook 840 G6

Device name	weihe01
Processor	Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz 1.99 GHz
Installed RAM	16.0 GB (15.8 GB usable)
Device ID	13ed4ffd-bd23-4ade-ad11-e53d9637ceb2
Product ID	00329-00000-00003-AA813
System type	64-bit operating system, x64-based processor
Pen and touch	No pen or touch input is available for this display

Rename this PC

Windows specifications

Edition	Windows 10 Enterprise
Version	1909
Installed on	1/21/2021
OS build	18363.1679

Figure 1 Computer Specification

3 Software Configuration

The laptop come with Windows 10 Operation System and Microsoft Office suit installed. There are a number of additional software pack are installed as below

- Python
- Anaconda
- Jupyter notebook

3.1 Python

Python was installed together with Anaconda with version 3.8.8 as Figure 2

```
(base) C:\Users\weihe>python -V
Python 3.8.8
```

Figure 2 Python version

3.2 Anaconda

Anaconda is the software downloaded from anaconda.com as Figure 3. it is a distribution of the Python and R programming languages for scientific computing

Download the software from anaconda.com and click the installation file for below Figure 3 to show

Figure 3 Anaconda download

After downloaded the Anaconda, I installed the software with Python package as Figure 4, Figure 5 and Figure 6.

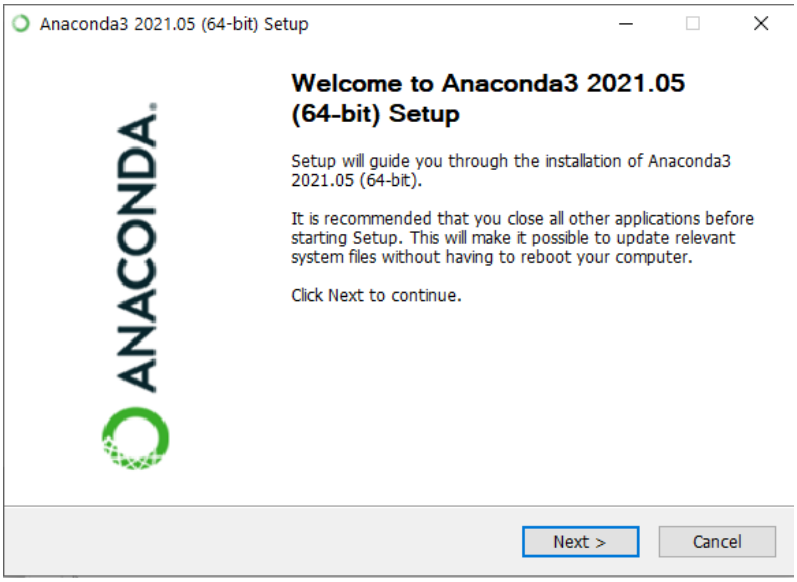


Figure 4 Anaconda installation

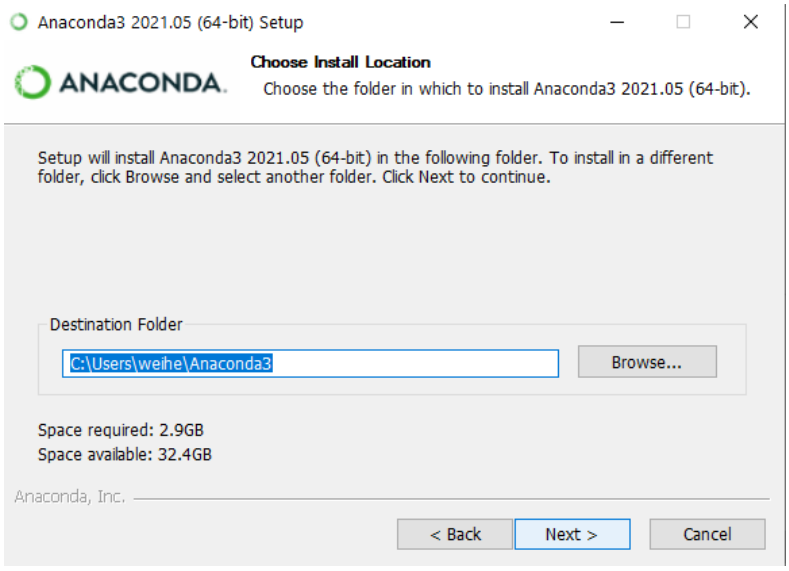


Figure 5 Anaconda installation with path selection

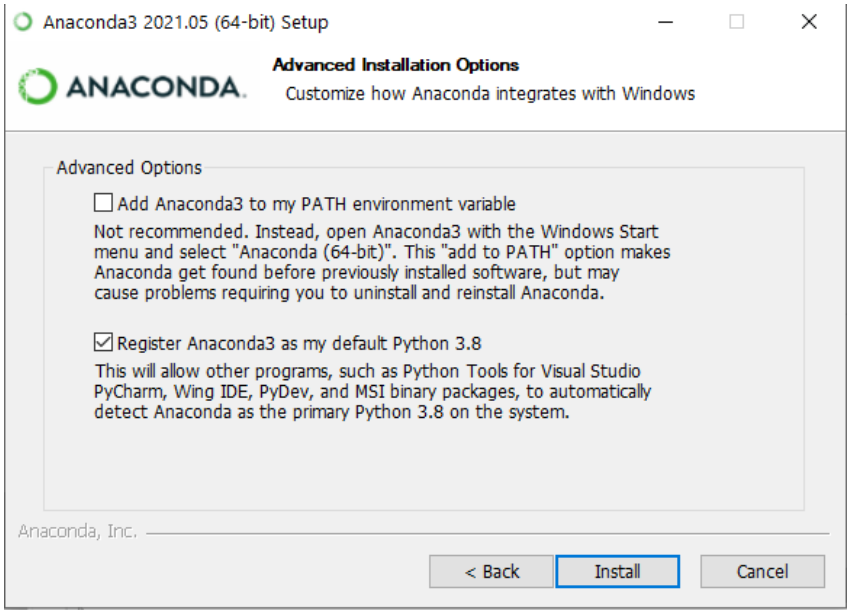


Figure 6 Anaconda Python 3.8 installation

3.3 Jupyter Notebook

Jupyter is a web-based interactive computational environment for creating Jupyter notebook documents. It is run under the Anaconda Interface

1. Open the Anaconda first as Figure 7

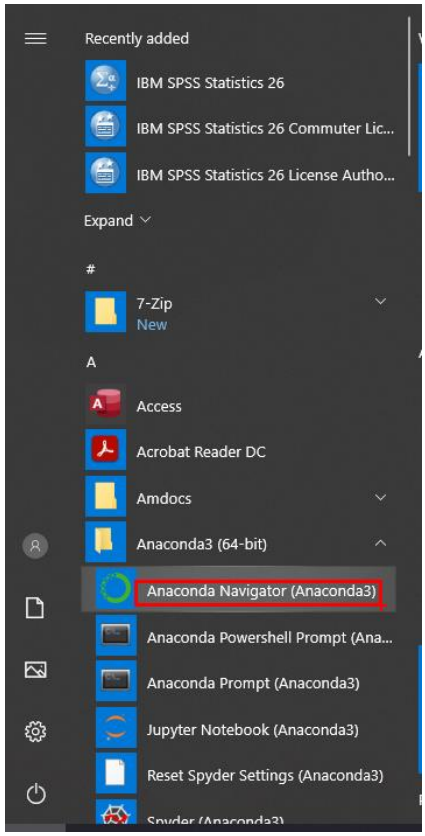


Figure 7 Open Anaconda

2. Run the jupyter notebook by click the Jupyter Notebook as Figure 8Figure 8

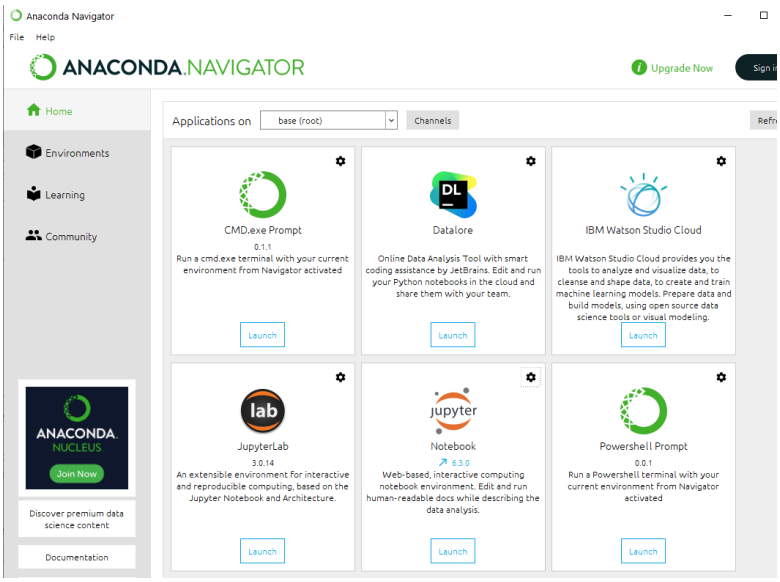


Figure 8 Jupyter notebook

3. Jupyter notebok open browser with http://localhost:8888 as Figure 9

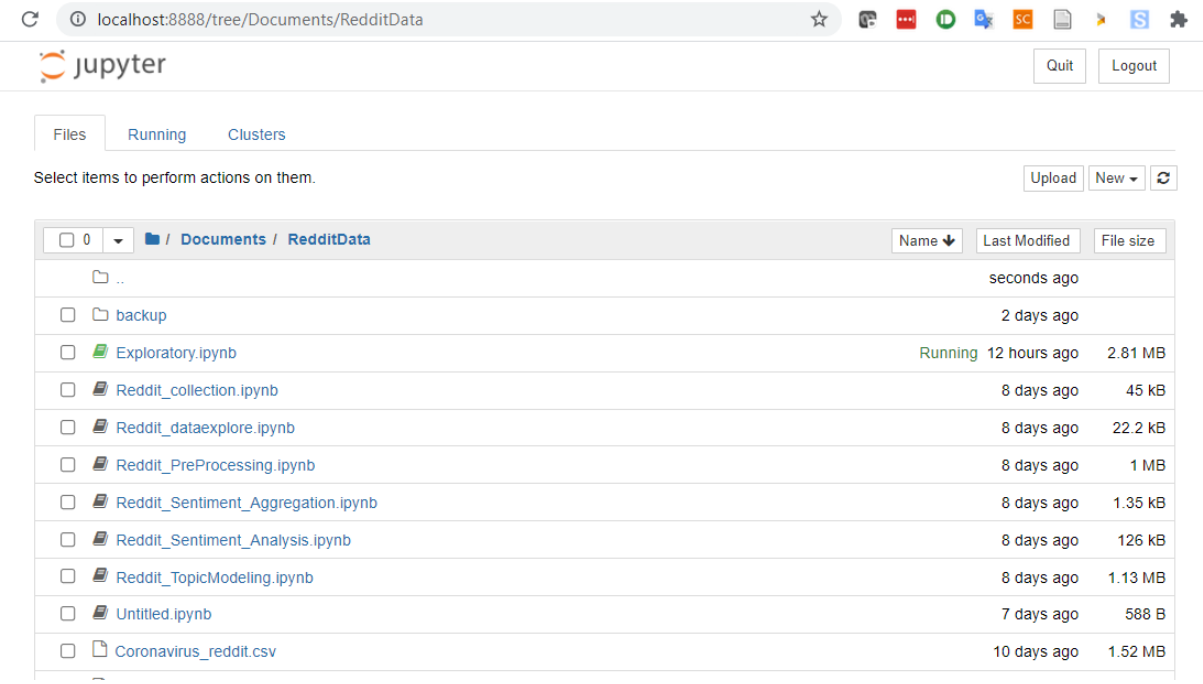


Figure 9 Jupyter web interface

3.4 IBM SPSS

IBM SPSS was used for data processing, running the regression model. NCI website has hosted the SPSS installer file for version 26. I download the installer to local computer hard drive then perform the installation as Figure 10 and Figure 11.

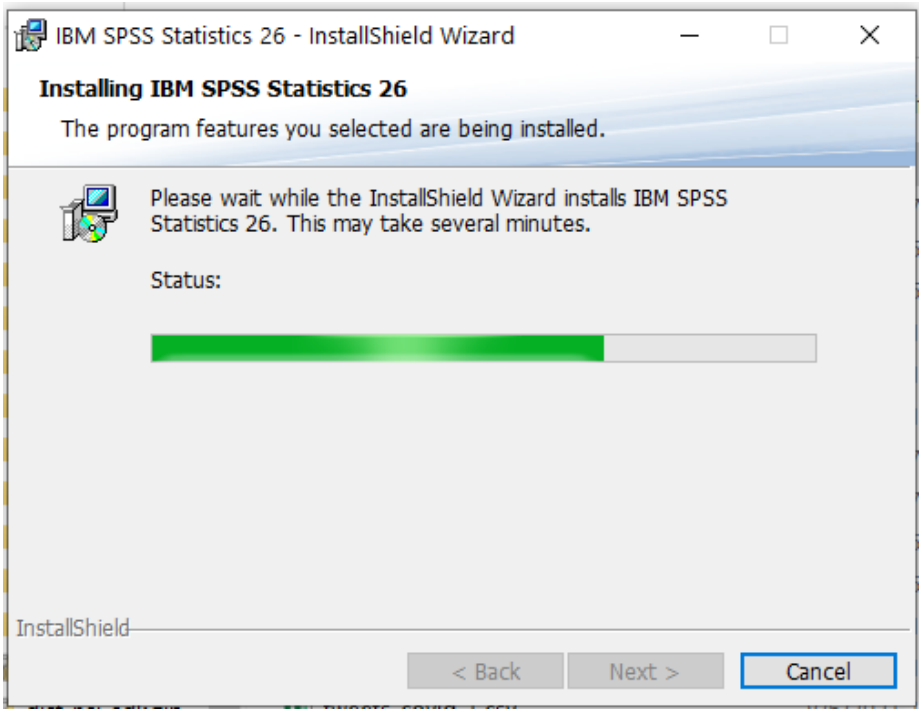


Figure 10 SPSS installation

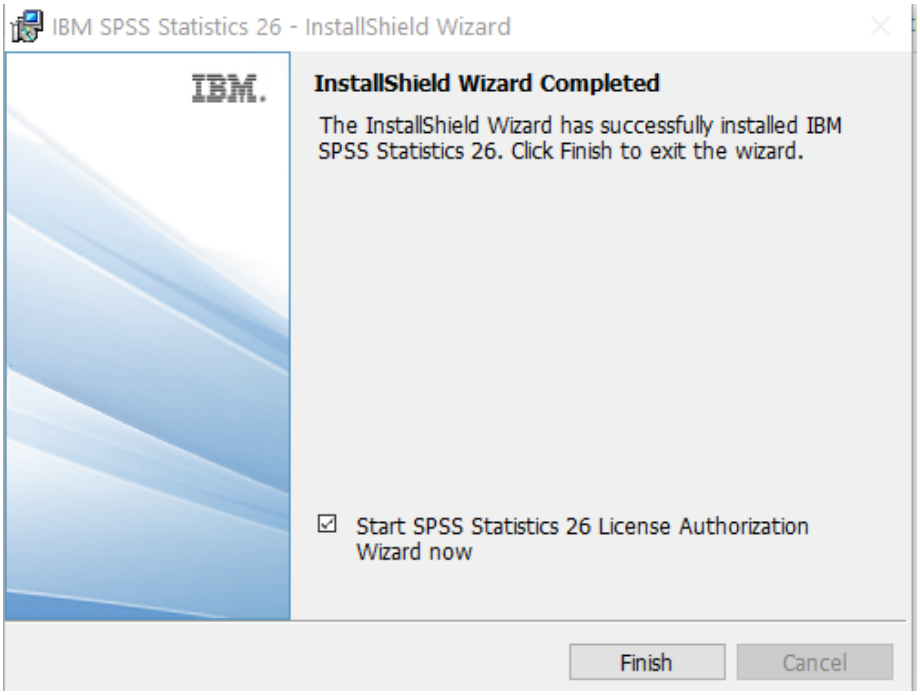


Figure 11 SPSS Installation completed

The SPSS authorization run automatically after the product installation as Figure 12, Provide the authorization code as Figure 13 and authorization completed as Figure 14.

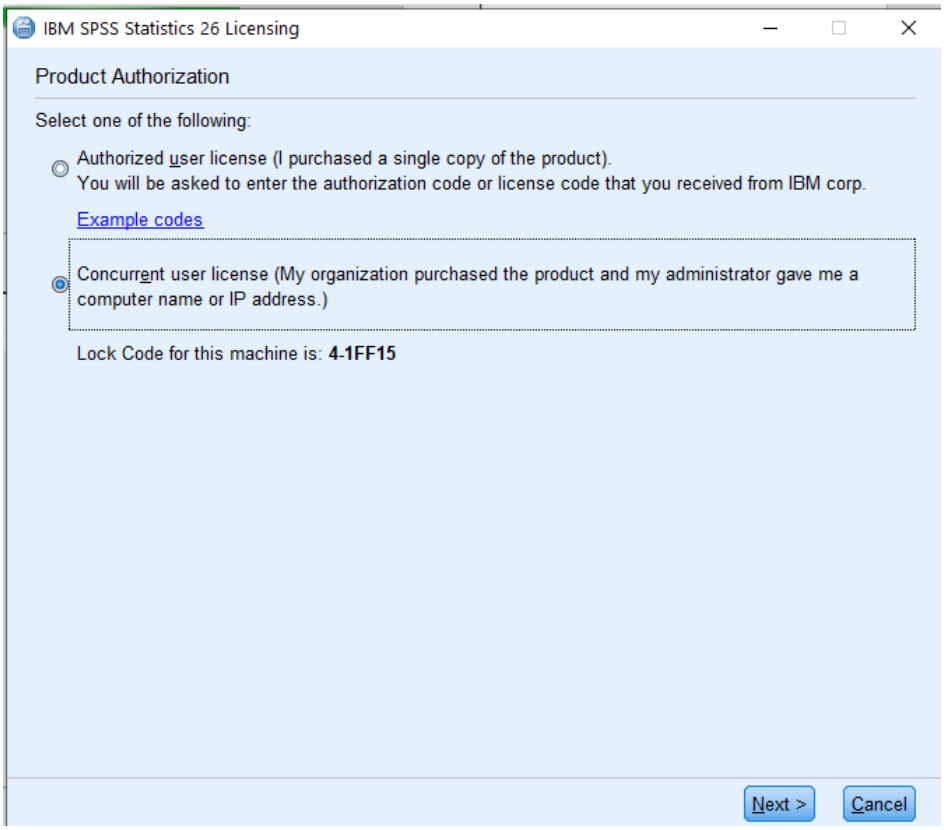


Figure 12 SPSS Product Authorization

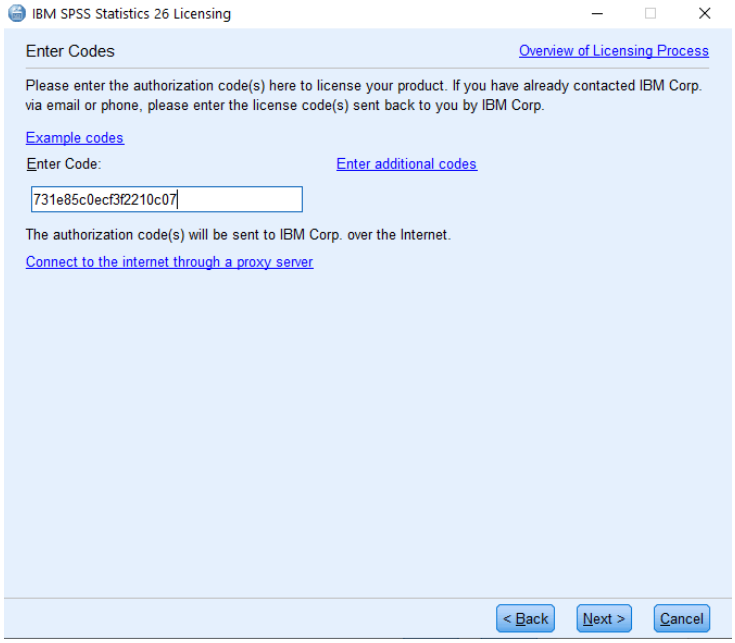


Figure 13 Provide Authorization Code

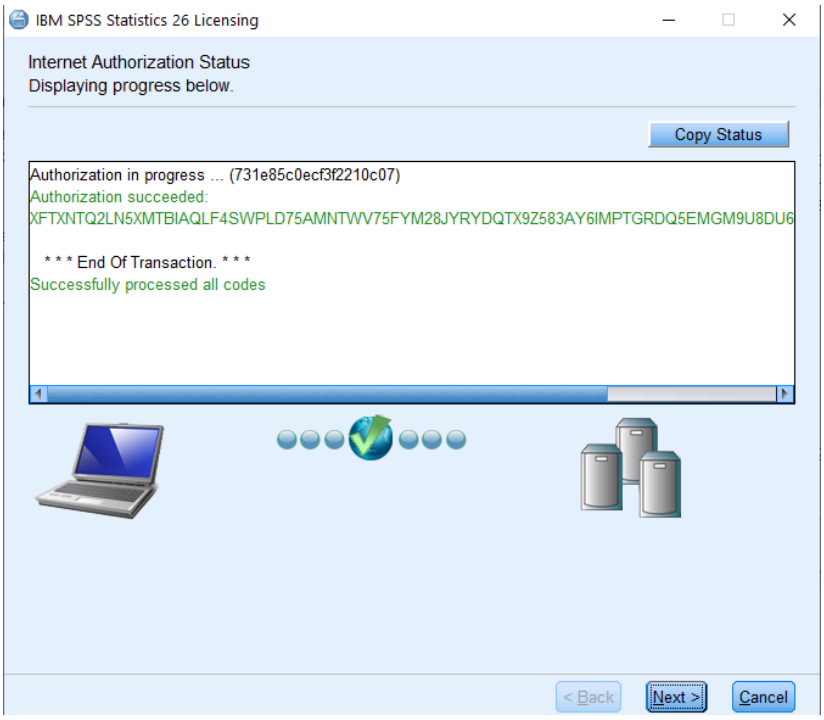


Figure 14 Authorization Successful

4 Datasets

Social media Dataset was collected by scraping the data from the twitter and reddit. Stock market Dataset collection was collected from the stock market website.

4.1 Data Creation

Data creation process are described in the below sections.

4.1.1 Scrape Twitter Data

Hashtag of twitter was searched using the Hastagify tool to find the best suitable hashtag for investigate this research topic as Figure 15

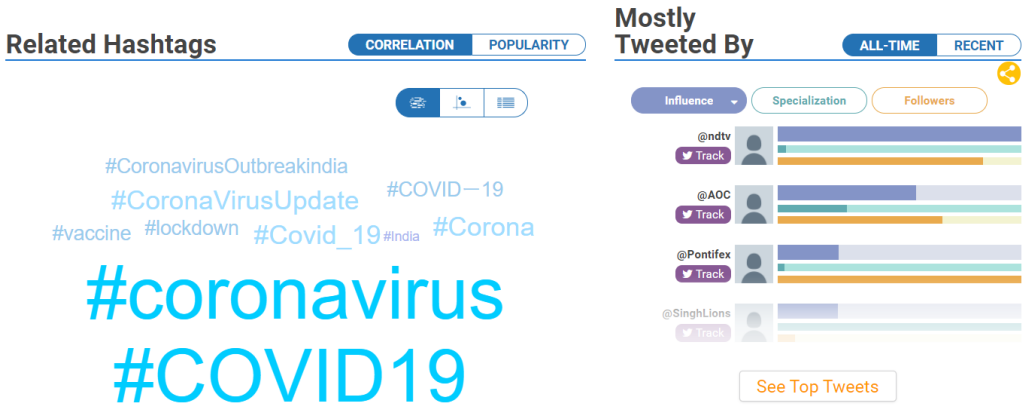


Figure 15 Hashtagify search tool

I created a twitter develop account and get the approval for the Academic Research which allows download 10 million tweets every 30 days as below Developer Portal in Figure 16

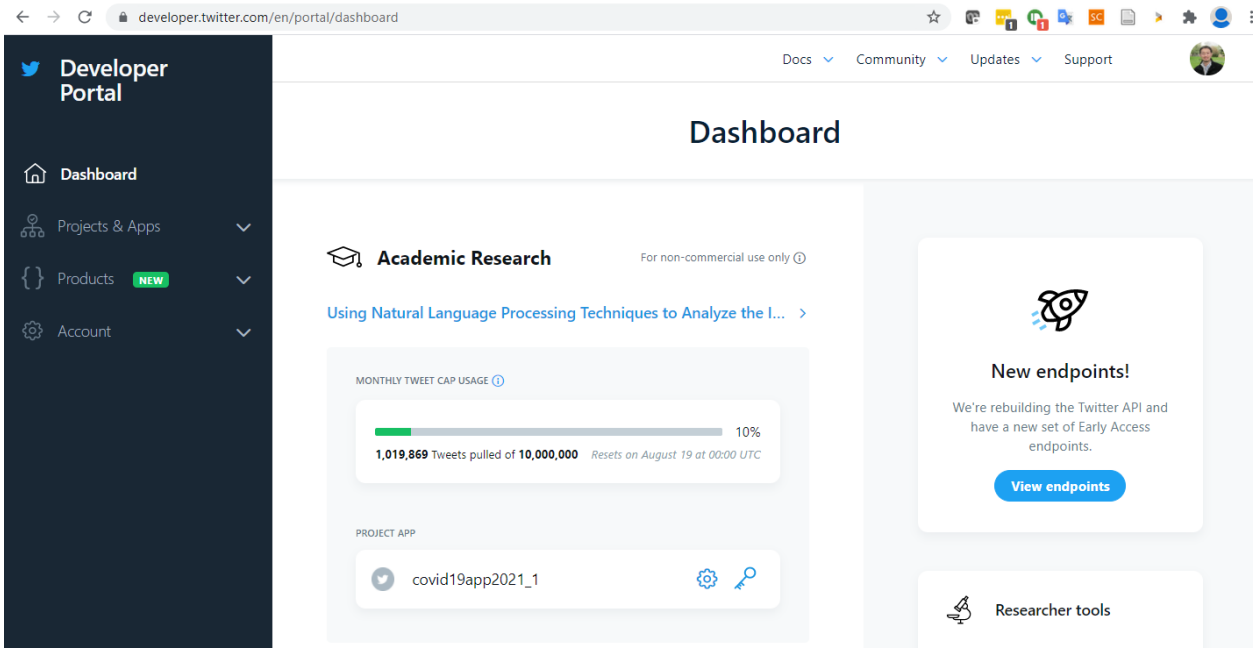


Figure 16 Twitter Developer Portal

I created a develop app under the twitter account as Figure 17.

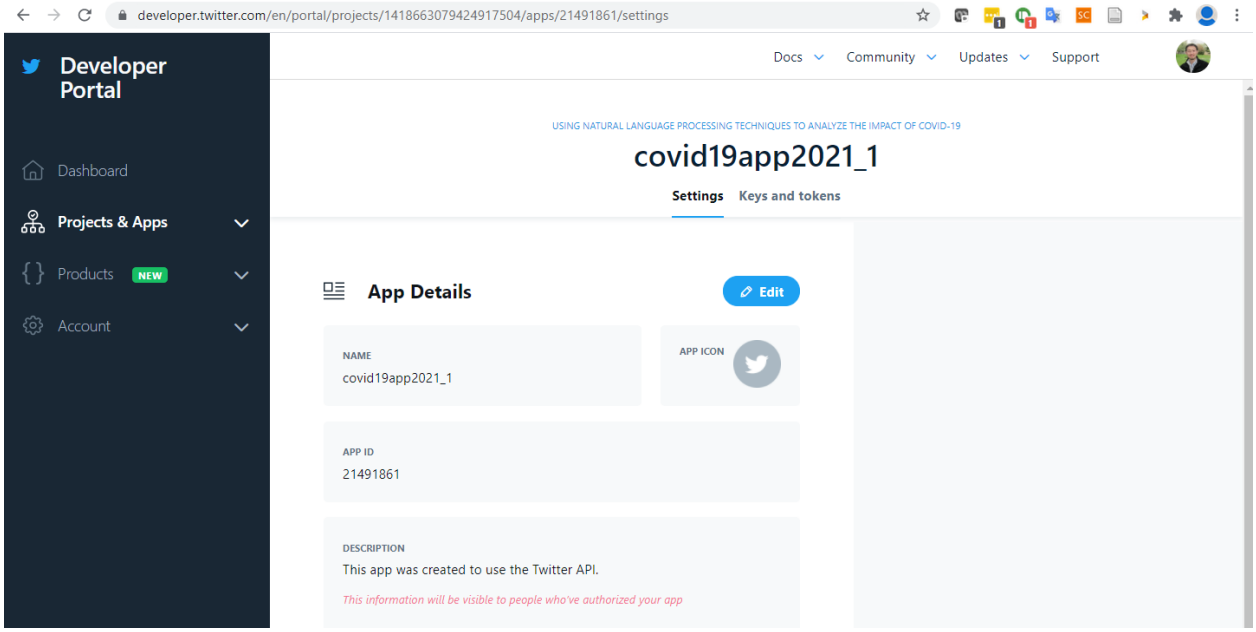


Figure 17 App creation

As the volume of tweets are large, I repeated the script 5 times to fetch the 5-month twitter data, each json file contains 1 month data. Twarc script is shown as Figure 18. Script logs run as Figure 19. After script is done, I collect 5-month tweet data in json format as Figure 20.

```python
from twarc import Twarc2, expansions
import datetime
import json

# Replace your bearer token below
client = Twarc2(bearer_token=
"AAAAAAAAAAAAAAAAAAAAAAJXwRwEAAAAAe%2FgCE6wrMj83iG%2F1DEQVBQGM8Cc%3DNMlmJGrq3EkxJD0dbwaWfFx9Wk4AowIXBfuNSAVWrKKQ12YhFC")



def main():
    # Specify the start time in UTC for the time period you want replies from
    start_time = datetime.datetime(2020, 1, 22, 0, 0, 0, 0, datetime.timezone.utc)

    # Specify the end time in UTC for the time period you want Tweets from
    end_time = datetime.datetime(2020, 2, 22, 0, 0, 0, 0, datetime.timezone.utc)


    # This is where we specify our query as discussed in module 5
    #query = "from:twitterdev"
    query = '#covid lang:en -is:retweet'

    # The search_all method call the full-archive search endpoint to get Tweets based on the query, start and end times
    search_results = client.search_all(query=query, start_time=start_time, end_time=end_time, max_results=100)


    # Twarc returns all Tweets for the criteria set above, so we page through the results

    file_name = 'tweets_covid_1.json'

    for page in search_results:
        # The Twitter API v2 returns the Tweet information and the user, media etc.  separately
        # so we use expansions.flatten to get all the information in a single JSON
        result = expansions.flatten(page)
        # We will open the file and append one JSON object per new line
        with open(file_name, 'a+') as filehandle:
            for tweet in result:
                filehandle.write('%s\n' % json.dumps(tweet))



if __name__ == "__main__":
    main()
```

Figure 18 Twarc script

```
(base) C:\Users\weihe\Documents>python twarc_project.py
rate limit exceeded: sleeping 586.1032757759094 secs
rate limit exceeded: sleeping 582.3726861476898 secs
rate limit exceeded: sleeping 580.1310176849365 secs
rate limit exceeded: sleeping 587.95143699646 secs
rate limit exceeded: sleeping 579.7003827095032 secs
rate limit exceeded: sleeping 586.2298331260681 secs
rate limit exceeded: sleeping 588.5079460144043 secs
rate limit exceeded: sleeping 585.1857266426086 secs
rate limit exceeded: sleeping 582.1188070774078 secs
rate limit exceeded: sleeping 578.5377037525177 secs
rate limit exceeded: sleeping 577.139200925827 secs
rate limit exceeded: sleeping 579.1963152885437 secs
rate limit exceeded: sleeping 583.7487273216248 secs
rate limit exceeded: sleeping 580.32887840271 secs
rate limit exceeded: sleeping 579.6935012340546 secs
rate limit exceeded: sleeping 566.9035663604736 secs
rate limit exceeded: sleeping 613.296749830246 secs
rate limit exceeded: sleeping 569.7559633255005 secs
rate limit exceeded: sleeping 572.3387174606323 secs
rate limit exceeded: sleeping 527.9719495773315 secs
rate limit exceeded: sleeping 533.4245765209198 secs
rate limit exceeded: sleeping 545.4147140979767 secs
```

Figure 19 Run Twarc script

```
(base) C:\Users\weihe\Documents\TweetData>dir *.json
 Volume in drive C is OSDisk
 Volume Serial Number is 3AE7-2B5E

 Directory of C:\Users\weihe\Documents\TweetData

08/05/2021  10:37 PM        29,565,435 tweets_covid_1.json
08/06/2021  12:36 AM     1,193,701,220 tweets_covid_2.json
08/06/2021  12:17 PM     3,205,631,618 tweets_covid_3.json
08/06/2021  05:30 PM     2,136,773,602 tweets_covid_4.json
08/06/2021  11:55 PM     1,526,286,261 tweets_covid_5.json
               5 File(s)  8,091,958,136 bytes
               0 Dir(s)  54,811,557,888 bytes free
```

Figure 20 Collected Twitter in json format

The academic research shows the ~1.7 million tweets was downloaded for this project as Figure 21

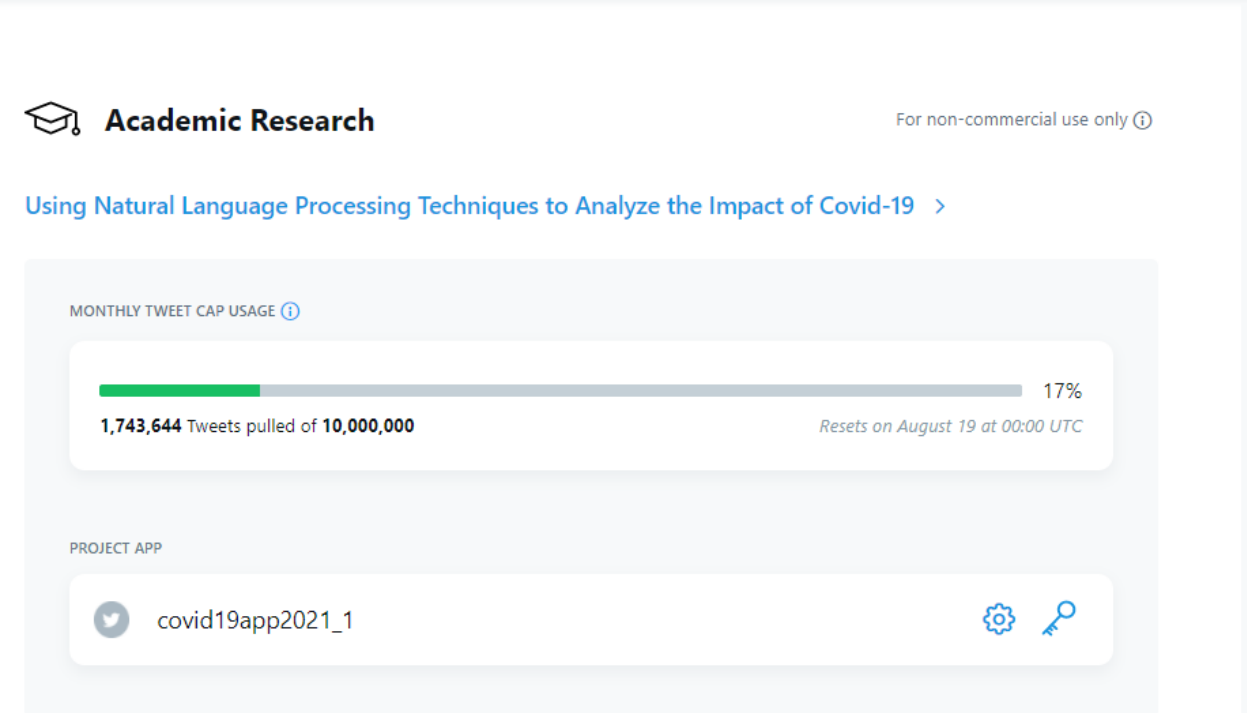


Figure 21 Academic Research panel

4.1.2 Formatting Twitter Data

I extracted the useful field and store them in the csv file for all the JSON file collected as Figure 22.

```
In [2]: def json_to_csv(json_file, csv_file):
            #data_json = open('tweets_covid_2.txt', mode='r').read() #reads in the JSON file into Python as a string
            #data_python = json.loads(data_json) #turns the string into a json Python object
            data_python = []
            for line in open(json_file, 'r'):
                data_python.append(json.loads(line))

            csv_out = open(csv_file, mode='w') #opens csv file
            writer = csv.writer(csv_out) #create the csv writer object

            fields = ['conversation_id','lang','timestamp', 'origin', 'author', 'rt', 'fav'] #field names
            writer.writerow(fields) #writes field

            for line in data_python:

                #writes a row and gets the fields from the json object
                #screen_name and followers/friends are found on the second level hence two get methods
                writer.writerow([line.get('conversation_id'),
                                 line.get('lang'),
                                 line.get('created_at'),
                                 line.get('text').encode('unicode_escape'), #unicode escape to fix emoji issue
                                 line.get('author_id'),
                                 line.get('public_metrics').get('retweet_count'),
                                 line.get('public_metrics').get('like_count')])

            csv_out.close()
```

Figure 22 Extract fields from JSON and store in CSV

I process all the JSON file and store them in CSV as Figure 23 Load all the CSV and merge them in one dataframe as Figure 24

```
In [4]: json_to_csv("tweets_covid_1.json","tweets_covid_1.csv" )
        json_to_csv("tweets_covid_2.json","tweets_covid_2.csv" )
        json_to_csv("tweets_covid_3.json","tweets_covid_3.csv" )
        json_to_csv("tweets_covid_4.json","tweets_covid_4.csv" )
        json_to_csv("tweets_covid_5.json","tweets_covid_5.csv" )
```

Figure 23 Process for all json files

```
In [6]: tweet1 = pd.read_csv("tweets_covid_1.csv")
        tweet2 = pd.read_csv("tweets_covid_2.csv")
        tweet3 = pd.read_csv("tweets_covid_3.csv")
        tweet4 = pd.read_csv("tweets_covid_4.csv")
        tweet5 = pd.read_csv("tweets_covid_5.csv")

In [7]: li = []
        li.append(tweet1)
        li.append(tweet2)
        li.append(tweet3)
        li.append(tweet4)
        li.append(tweet5)

In [8]: frame = pd.concat(li, axis=0, ignore_index=True)

In [9]: frame.to_csv(r'C:\Users\weihe\Documents\TweetData\tweet_aggregated.csv', index =
```

Figure 24 Read in Pandas dataframe

I back up the text field and output the new structure as Figure 25 and Figure 26 .

```
In [12]: # duplicate origin field for processing.
         frame["text"] = frame["origin"]
         frame.head()
```

Out[12]:

	conversation_id	lang	timestamp	origin	author	rt	fav	text
0	1231005111071100929	en	2020-02-21T23:57:44.000Z	b'Italy \\U0001f1ee\\U0001f1f9 reports first I...	2985110557	2	1	b'Italy \\U0001f1ee\\U0001f1f9 reports first I...
1	1231005062165458945	en	2020-02-21T23:57:32.000Z	b'#Coronavirus quarantine in a Hubu Village, S...	1202960464549838848	1	2	b'#Coronavirus quarantine in a Hubu Village, S...
2	1231004398811275264	en	2020-02-21T23:54:54.000Z	b'Italy. I love this country. #coronavirusital...	1221955502340558848	0	3	b'Italy. I love this country. #coronavirusital...
3	1230866203947798530	en	2020-02-21T23:53:24.000Z	b'@JoannaB36464141 @nicolamlow @alexvespi @Ada...	1093695069176000512	0	2	b'@JoannaB36464141 @nicolamlow @alexvespi @Ada...
4	1231003954806501376	en	2020-02-21T23:53:08.000Z	b'@WHO \\nThis is a #pandemic \\n#CoronavirusO...	723319704710844417	0	0	b'@WHO \\nThis is a #pandemic \\n#CoronavirusO...

Figure 25 backup the text fields

```
In [13]: print(frame.shape)
         print(frame.columns)

         (1634067, 8)
         Index(['conversation_id', 'lang', 'timestamp', 'origin', 'author', 'rt', 'fav',
                'text'],
               dtype='object')
```

Figure 26 Size of the dataframe

Display the data and exam the structure is done as Figure 27, Figure 28, Figure 29, Figure 30, Figure 31Figure 32 before save the it to CSV file as Figure 33

```
In [14]: #Describe statistics with 'number' for numeric variables
         frame.describe(include=['number'])
```

Out[14]:

	conversation_id	author	rt	fav
count	1.634067e+06	1.634067e+06	1.634067e+06	1.634067e+06
mean	1.252350e+18	3.601813e+17	2.594112e+00	8.015488e+00
std	1.314828e+16	5.087070e+17	1.538563e+02	3.182827e+02
min	1.837335e+09	5.090000e+02	0.000000e+00	0.000000e+00
25%	1.243971e+18	1.599892e+08	0.000000e+00	0.000000e+00
50%	1.251206e+18	1.588390e+09	0.000000e+00	0.000000e+00
75%	1.260615e+18	9.171150e+17	1.000000e+00	2.000000e+00
max	1.274855e+18	1.274780e+18	1.404620e+05	1.896340e+05

Figure 27 Statistics of dataframe

```
In [15]: #Describe statistics with 'object' for string variables
         frame.describe(include=['object'])
```

Out[15]:

	lang	timestamp	origin	text
count	1634067	1634067	1634067	1634067
unique	28	1442803	1603448	1603448
top	en	2020-06-10T05:02:52.000Z	b'To fight #COVID-19 we need to work together ...	b'To fight #COVID-19 we need to work together ...
freq	1630290	17	1018	1018

Figure 28 Describe string variable

```
In [16]: # check is there any missing values in dataframe
         frame.isnull()
```

Out[16]:

	conversation_id	lang	timestamp	origin	author	rt	fav	text
0	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...
1634062	False	False	False	False	False	False	False	False
1634063	False	False	False	False	False	False	False	False
1634064	False	False	False	False	False	False	False	False
1634065	False	False	False	False	False	False	False	False
1634066	False	False	False	False	False	False	False	False

1634067 rows × 8 columns

Figure 29 Check for Null object

```
In [18]: print(frame.dtypes)

conversation_id     int64
lang               object
timestamp          object
origin             object
author              int64
rt                  int64
fav                 int64
text               object
dtype: object
```

Figure 30 Print dateframe type

```
In [19]: # Only keep english tweet
         frame = frame[frame["lang"] == 'en']
         frame.shape

Out[19]: (1630290, 8)
```

Figure 31 Recheck the size of the dataframe

```
In [21]: # print period of tweets
         datemin = frame.timestamp.min()
         datemax = frame.timestamp.max()
         print('Collected tweets from', datemin, 'To', datemax)

         Collected tweets from 2020-02-04T18:23:21.000Z To 2020-06-21T23:59:59.000Z
```

Figure 32 Data collection range

```
In [22]: frame.to_csv(r'C:\Users\weihe\Documents\TweetData\tweet_aggregated.csv', index = Fa
```

Figure 33 Store Dataframe

4.1.3 Scrape Reddit Data

I chose the subreddit data by search popular hash tag. I selected the 2 subreddit as shown in Figure 34, Figure 35 and Figure 36

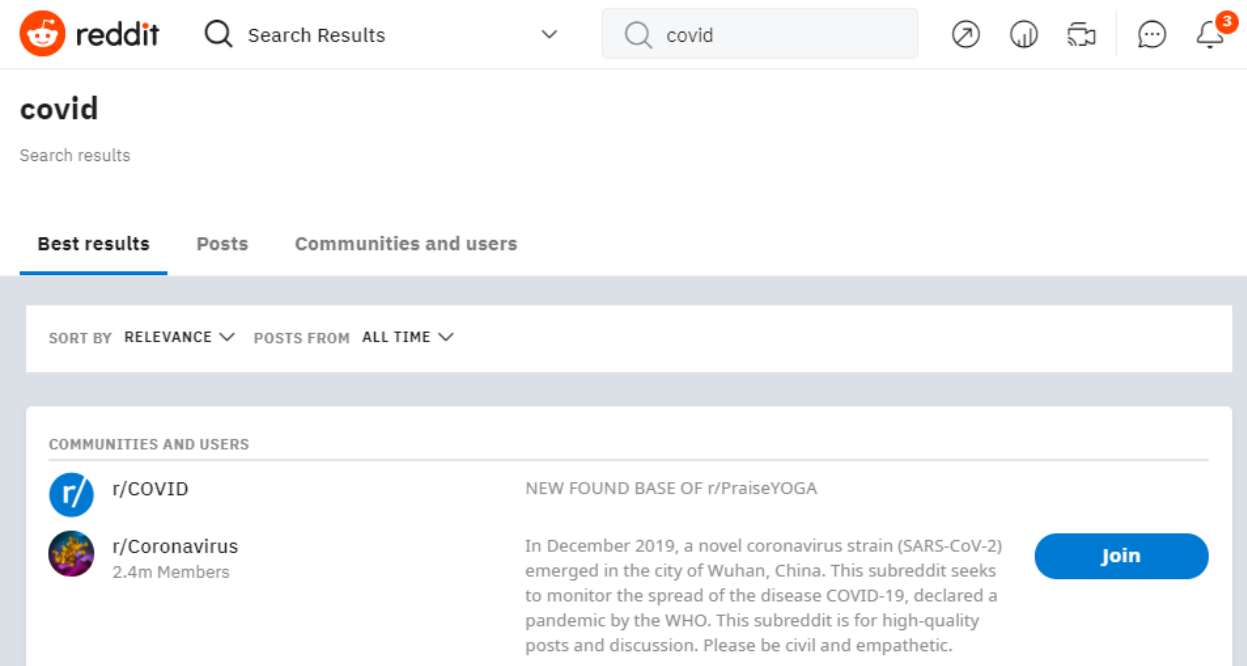


Figure 34 Search for popular subreddit

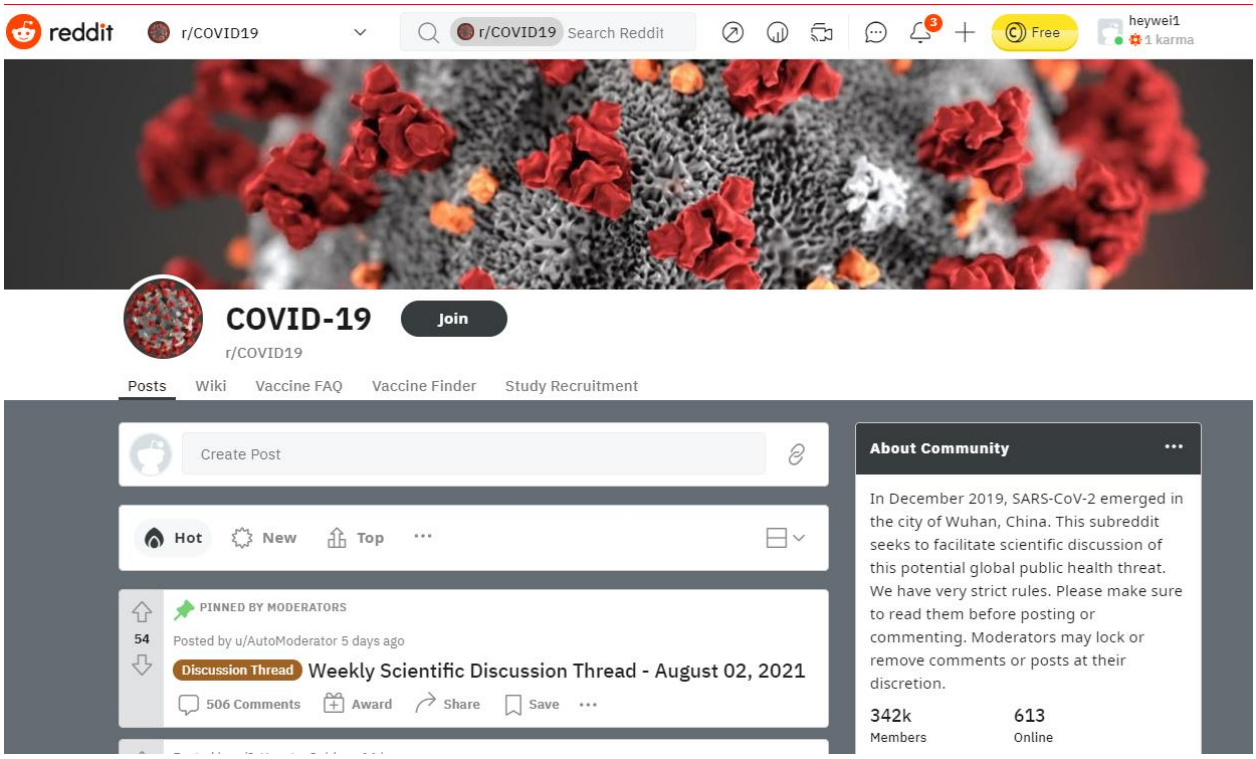


Figure 35 COVID19 subreddit

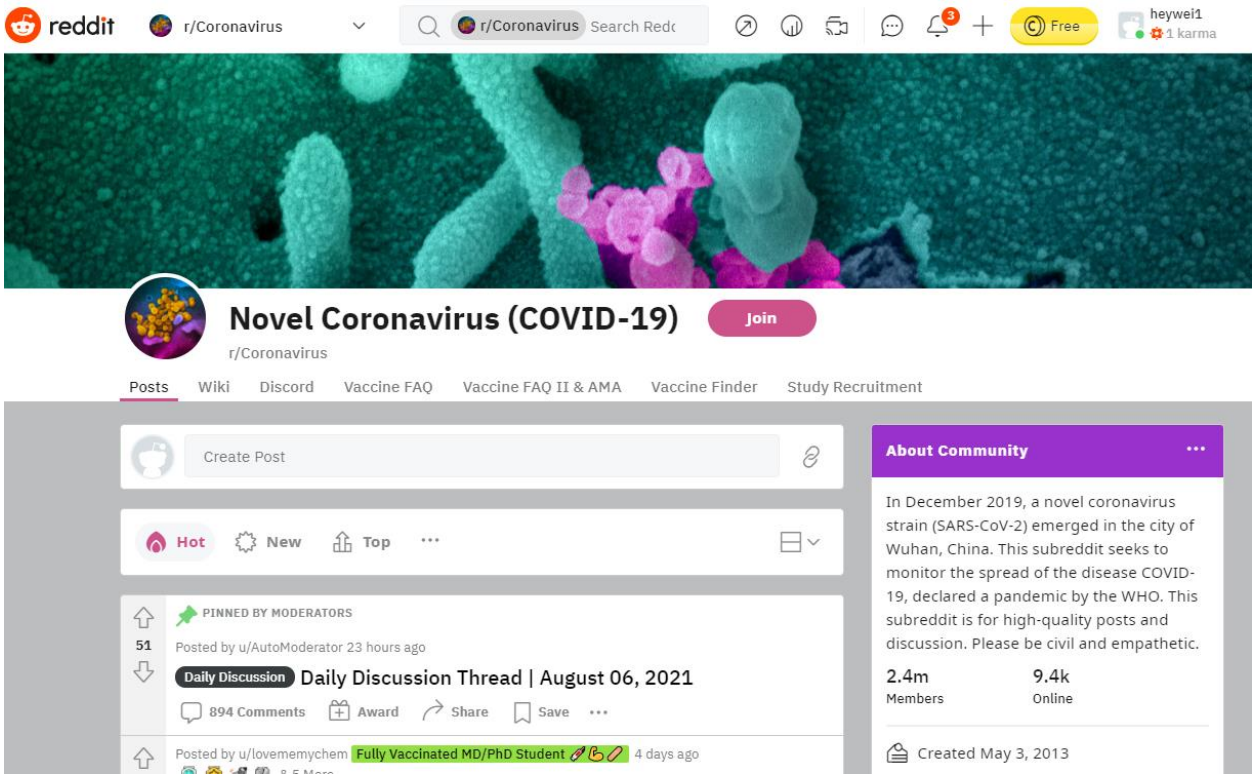


Figure 36 CoronaVirus subreddit

I used the PSAW wrapper to the scrape the Reddit data, it can use the pushshift.io to search historic records with data range provided as Figure 37.

Docs » PSAW: Python Pushshift.io API Wrapper (for comment/submission search)

Edit on GitHub

PSAW: Python Pushshift.io API Wrapper (for comment/submission search)

Installation

```
pip install psaw
```

Description

A minimalist wrapper for searching public reddit comments/submissions via the pushshift.io API.

Pushshift is an extremely useful resource, but the API is poorly documented. As such, this API wrapper is currently designed to make it easy to pass pretty much any search parameter the user wants to try.

Although it is not necessarily reflective of the current status of the API, you should attempt to familiarize yourself with the Pushshift API documentation to better understand what search arguments are likely to work.

Figure 37 PSAW library to access Pushshift.io API

Reddit account was created to scrape the subreddit as Figure 38

personal use script

WSbO8uEVRGG5FK6DSe8yRg

personal use for pulling recent data for analysis

secret jBMLgB02nKNcLSJjR17e1pwfATEk1g

name

description

about url

redirect uri

update app

developers • heywei1 (that's you!) remove

add developer:

Figure 38 Reddit personal access key

To scrape the reddit, I started by import python library as Figure 39

```
In [1]: import pandas as pd
        import praw
        from praw.models import MoreComments
        import pandas as pd
        import requests #Pushshift accesses Reddit via an url so this is needed
        import json #JSON manipulation
        import csv #To Convert final table into a csv file to save to your machine
        import time
        import datetime
```

Figure 39 Import library

I used the PRSW library to collect subreddit Figure 40 with function defined in Figure 41 and function to collect subreddit data in Figure 42.

```
In [2]: reddit = praw.Reddit(user_agent="Comment Extraction (by /u/heywei1)",
                             client_id="WSbO8uEVRGG5FK6DSe8yRg", client_secret="jBMLgB02nKNcLSJjR17e1pwfATEk1g")
```

```
In [3]: url = "https://www.reddit.com/r/UpliftingNews/comments/lemy1b/student_who_made_30k_from_gamestop_donates_games/"
        submission = reddit.submission(url=url)
```

```
In [4]: posts = []
        for top_level_comment in submission.comments[1:]:
            if isinstance(top_level_comment, MoreComments):
                continue
            posts.append(top_level_comment.body)
        posts = pd.DataFrame(posts,columns=["body"])
```

```
In [5]: posts
```

Out[5]:

	body
0	why be so cynical? Regardless of his motives,...
1	I want to be annoyed by this repost, but I can...
2	I love the positive things that GME "winners" ...
3	As a fellow gamer this guy is who I aspire to ...

Figure 40 Collect Subreddit using PRSW library

```
In [12]: def getPushshiftData(query, after, before, sub):
             #Build URL
             url = 'https://api.pushshift.io/reddit/search/submission/?title='+str(query)+'&size=1000&after='+st
             #Print URL to show user
             print(url)
             #Request URL
             r = requests.get(url)
             #Load JSON data from webpage into data variable
             data = json.loads(r.text)
             #return the data element which contains all the submissions data
             return data['data']
```

Figure 41 Function defined

```
In [13]: #This function will be used to extract the key data points from each JSON result
         def collectSubData(subm):
             #subData was created at the start to hold all the data which is then added to our global subStats d
             subData = list() #list to store data points
             title = subm['title']
             url = subm['url']
             #flairs are not always present so we wrap in try/except
             try:
                 flair = subm['link_flair_text']
             except KeyError:
                 flair = "NaN"
             author = subm['author']
             sub_id = subm['id']
             score = subm['score']
             created = datetime.datetime.fromtimestamp(subm['created_utc']) #1520561700.0
             numComms = subm['num_comments']
             permalink = subm['permalink']

             #Put all data points into a tuple and append to subData
             subData.append((sub_id,title,url,author,score,created,numComms,permalink,flair))
             #Create a dictionary entry of current submission data and store all data related to it
             subStats[sub_id] = subData
```

Figure 42 Collect subreddit data function

I defined the time period to collect the reddit as Figure 43 and Figure 44

Conversion results - Epoch to date

Epoch date	Human-readable date (GMT)
1579651200	2020-01-22 00:00:00
1592784000	2020-06-22 00:00:00

Figure 43 Convert the time to Epoch time

```
In [18]: #Create your timestamps and queries for your search URL
         #https://www.unixtimestamp.com/index.php > Use this to create your timestamps
         after = "1579651200" #Submissions after this timestamp (1579651200 = 22 Jan 2020)
         before = "1592784000" #Submissions before this timestamp (1592784000 = 22 Jun 2020)
         query = "covid" #Keyword(s) to look for in submissions
         sub = "COVID19" #Which Subreddit to search in

         #subCount tracks the no. of total submissions we collect
         subCount = 0
         #subStats is the dictionary where we will store our data.
         subStats = {}
```

Figure 44 Define collection time period

```
In [19]: # We need to run this function outside the loop first to get the updated after variable
         data = getPushshiftData(query, after, before, sub)
         # Will run until all posts have been gathered i.e. When the length of data variable = 0
         # from the 'after' date up until before date
         while len(data) > 0: #The length of data is the number submissions (data[0], data[1] etc), once it hit
             for submission in data:
                 collectSubData(submission)
                 subCount+=1
             # Calls getPushshiftData() with the created date of the last submission
             print(len(data))
             print(str(datetime.datetime.fromtimestamp(data[-1]['created_utc'])))
             #update after variable to last created date of submission
             after = data[-1]['created_utc']
             #data has changed due to the new after variable provided by above code
             data = getPushshiftData(query, after, before, sub)

         print(len(data))

         https://api.pushshift.io/reddit/search/submission/?title=covid&size=1000&after=1579651200&before=15
         92784000&subreddit=COVID19
         100
         2020-02-25 10:28:17
         https://api.pushshift.io/reddit/search/submission/?title=covid&size=1000&after=1582626497&before=15
         92784000&subreddit=COVID19
         100
```

Figure 45 Define the get the Pushshift wrapper

I printed the statistics after data collection as Figure 46 before storing it in CSV as Figure 47

```
In [20]: print(str(len(subStats)) + " submissions have added to list")
         print("1st entry is:")
         print(list(subStats.values())[0][0][1] + " created: " + str(list(subStats.values())[0][0][5]))
         print("Last entry is:")
         print(list(subStats.values())[-1][0][1] + " created: " + str(list(subStats.values())[-1][0][5]))

         5326 submissions have added to list
         1st entry is:
         Found something in "The Lancet" regarding cytokine storms as a symptom of Covid-19 (Article Date: Feb
         ruary 03, 2020) https://www.thelancet.com/journals/lanres/article/PIIS2213-2600(20)30056-4/fulltext c
         reated: 2020-02-12 03:28:58
         Last entry is:
         COVID-19 Evidence is lacking for 2 meter distancing created: 2020-06-22 00:02:25
```

Figure 46 Print stats after the collection

```
In [21]: def updateSubs_file():
             upload_count = 0
             #location = "\\Reddit Data\\" >> If you're running this outside of a notebook you'll need this to c
             print("input filename of submission file, please add .csv")
             filename = input() #This asks the user what to name the file
             file = filename
             with open(file, 'w', newline='', encoding='utf-8') as file:
                 a = csv.writer(file, delimiter=',')
                 headers = ["Post ID","Title","Url","Author","Score","Publish Date","Total No. of Comments","Per
                 a.writerow(headers)
                 for sub in subStats:
                     a.writerow(subStats[sub][0])
                     upload_count+=1

                 print(str(upload_count) + " submissions have been uploaded")
         updateSubs_file()

         input filename of submission file, please add .csv
         COVID_reddit.csv
         5326 submissions have been uploaded
```

Figure 47 store the COVID_reddit.csv

Same collection procedure was used to collect second subreddit date as Figure 48, Figure 49, Figure 50 and Figure 51

```
In [23]: #Create your timestamps and queries for your search URL
         #https://www.unixtimestamp.com/index.php > Use this to create your timestamps
         after = "1579651200" #Submissions after this timestamp (1577836800 = 01 Jan 20)
         before = "1592784000" #Submissions before this timestamp (1607040000 = 04 Dec 20)
         query = "Corona" #Keyword(s) to look for in submissions
         sub = "Coronavirus" #Which Subreddit to search in

         #subCount tracks the no. of total submissions we collect
         subCount = 0
         #subStats is the dictionary where we will store our data.
         subStats = {}
```

Figure 48 Define collection period

```
In [24]: # We need to run this function outside the loop first to get the updated after variable
data = getPushshiftData(query, after, before, sub)
# Will run until all posts have been gathered i.e. When the length of data variable = 0
# from the 'after' date up until before date
while len(data) > 0: #The length of data is the number submissions (data[0], data[1] etc), once it hits
    for submission in data:
        collectSubData(submission)
        subCount+=1
    # Calls getPushshiftData() with the created date of the last submission
    print(len(data))
    print(str(datetime.datetime.fromtimestamp(data[-1]['created_utc'])))
    #update after variable to last created date of submission
    after = data[-1]['created_utc']
    #data has changed due to the new after variable provided by above code
    data = getPushshiftData(query, after, before, sub)

print(len(data))
```

```
https://api.pushshift.io/reddit/search/submission/?title=Corona&size=1000&after=1579651200&before=1
592784000&subreddit=Coronavirus
100
2020-01-28 09:32:39
https://api.pushshift.io/reddit/search/submission/?title=Corona&size=1000&after=1580203959&before=1
592784000&subreddit=Coronavirus
100
```

Figure 49 Define function for get the PushShift data

```
In [25]: print(str(len(subStats)) + " submissions have added to list")
print("1st entry is:")
print(list(subStats.values())[0][0][1] + " created: " + str(list(subStats.values())[0][0][5]))
print("Last entry is:")
print(list(subStats.values())[-1][0][1] + " created: " + str(list(subStats.values())[-1][0][5]))
```

```
5326 submissions have added to list
1st entry is:
Trump will fight against corona virus created: 2020-01-22 06:29:40
Last entry is:
BREAKING NEWS DONALD TRUMP GETS CORONA VIRUS FROM OKLAHOMA RALLY created: 2020-06-21 23:39:54
```

Figure 50 print statistics

```
In [26]: def updateSubs_file():
    upload_count = 0
    #location = "\\Reddit Data\\" >> If you're running this outside of a notebook you'll need this to c
    print("input filename of submission file, please add .csv")
    filename = input() #This asks the user what to name the file
    file = filename
    with open(file, 'w', newline='', encoding='utf-8') as file:
        a = csv.writer(file, delimiter=',')
        headers = ["Post ID","Title","Url","Author","Score","Publish Date","Total No. of Comments","Per
        a.writerow(headers)
        for sub in subStats:
            a.writerow(subStats[sub][0])
            upload_count+=1

        print(str(upload_count) + " submissions have been uploaded")
updateSubs_file()
```

```
input filename of submission file, please add .csv
Coronavirus_reddit.csv
5326 submissions have been uploaded
```

Figure 51 Store the data to Coronavirus_reddit.csv file

4.1.4 Data Explore

I used the python to explore the date I collected. I imported library and load data as Figure 52

```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns

In [2]: covid = pd.read_csv("COVID_reddit.csv")

In [3]: corona = pd.read_csv("Coronavirus_reddit.csv")

In [4]: li = []
        li.append(covid)
        li.append(corona)

In [5]: frame = pd.concat(li, axis=0, ignore_index=True)
```

Figure 52 Create dateframe with COVID and Coronavirus subreddit

I checked the data and reorganicse the column as Figure 53, Figure 54 and Figure 55. Data was described in Figure 56

```
In [6]: frame.head()
Out[6]:
```

	Post ID	Title	Url	Author	Score	Publish Date	Total No. of Comments	Permali
0	f2kpf6	Found something in "The Lancet" regarding cyto...	https://www.reddit.com/r/COVID19/comments/f2kp...	ginpanse	1	2020-02-12 03:28:58	0	/r/COVID19/comments/f2kpf6/found_something_in_
1	f2m2cv	Any info on how COVID-19 affect pets/domestic ...	https://www.reddit.com/r/COVID19/comments/f2m2...	mhkk001	1	2020-02-12 05:13:07	5	/r/COVID19/comments/f2m2cv/any_info_on_how_co

Figure 53 Check Data

```
In [7]: frame["text"] = frame["Title"]

In [9]: frame = frame.drop('Permalink', 1)
        frame = frame.drop('Flair', 1)
        frame = frame.drop('Author', 1)
        frame = frame.drop('Url', 1)
```

Figure 54 Organize column

```
In [10]: frame.rename(columns = {'Title':'origin'}, inplace = True)
         frame.rename(columns = {'Publish Date':'timestamp'}, inplace = True)
         frame.rename(columns = {'Total No. of Comments':'comments'}, inplace = True)
         frame.rename(columns = {'Post ID':'id'}, inplace = True)
```

```
In [11]: print(frame.shape)
         print(frame.columns)

         (10652, 6)
         Index(['id', 'origin', 'Score', 'timestamp', 'comments', 'text'], dtype='object')
```

Figure 55 Check the data

```
In [12]: #Describe statistics with 'number' for numeric variables
         frame.describe(include=['number'])
```

Out[12]:

	Score	comments
count	10652.000000	10652.000000
mean	3.361998	12.568813
std	41.889694	53.358567
min	0.000000	0.000000
25%	1.000000	0.000000
50%	1.000000	1.000000
75%	1.000000	4.000000
max	2773.000000	1289.000000

Figure 56 Describe the data

I further checked the data for null check and data range before store in csv file as Figure 57, Figure 58, Figure 59, Figure 60, Figure 61 and Figure 62

```
In [13]: #Describe statistics with 'object' for string variables
         frame.describe(include=['object'])
```

Out[13]:

	id	origin	timestamp	text
count	10652	10652	10652	10652
unique	10652	10189	10643	10189
top	f2kpf6	Corona	2020-05-02 02:23:15	Corona
freq	1	14	3	14

Figure 57 describe the object

```
In [14]: # check is there any missing values in dataframe
         frame.isnull()
```

Out[14]:

	id	origin	Score	timestamp	comments	text
0	False	False	False	False	False	False
1	False	False	False	False	False	False
2	False	False	False	False	False	False
3	False	False	False	False	False	False
4	False	False	False	False	False	False
...	...	...	...	...	...	...
10647	False	False	False	False	False	False
10648	False	False	False	False	False	False
10649	False	False	False	False	False	False
10650	False	False	False	False	False	False
10651	False	False	False	False	False	False

10652 rows × 6 columns

Figure 58 Null check

```
In [15]: frame.isnull().sum()

Out[15]: id           0
         origin       0
         Score        0
         timestamp    0
         comments     0
         text         0
         dtype: int64
```

Figure 59 Null check statistics

```
In [16]: print(frame.dtypes)

         id           object
         origin       object
         Score         int64
         timestamp    object
         comments      int64
         text         object
         dtype: object
```

Figure 60 Check the data types

```
In [18]: # print period of tweets
         datemin = frame.timestamp.min()
         datemax = frame.timestamp.max()
         print('Collected tweets from', datemin, 'To', datemax)

         Collected tweets from 2020-01-22 06:29:40 To 2020-06-22 00:02:25
```

Figure 61 print data range

```
In [19]: frame.to_csv(r'C:\Users\weihe\Documents\RedditData\reddit_aggregated.csv', index = False,
```

Figure 62 Store the Data

4.2 Pre-processing

4.2.1 Twitter Data

Below steps defined how the twitter data is pre-processed.
I loaded the library and data with some initial exploration of data as Figure 63 Figure 64 .

```
In [1]: import json
        import csv
        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
        import os
        import emoji
        import re
        import nltk
        from nltk.stem import WordNetLemmatizer
```

Figure 63 Import library

```
In [2]: frame = pd.read_csv("tweet_aggregated.csv")

In [3]: frame.shape

Out[3]: (1630290, 8)

In [4]: print(frame.dtypes)

        conversation_id     int64
        lang               object
        timestamp          object
        origin             object
        author              int64
        rt                  int64
        fav                 int64
        text               object
        dtype: object
```

Figure 64 Print general information

I installed and loaded the tweet-preprocessor packet for clean the tweets as Figure 65 and Figure 66

```
(base) C:\Users\weihe>pip install tweet-preprocessor
Collecting tweet-preprocessor
  Downloading tweet_preprocessor-0.6.0-py3-none-any.whl (27 kB)
Installing collected packages: tweet-preprocessor
Successfully installed tweet-preprocessor-0.6.0
```

Figure 65 Download Python library

```
In [5]: import preprocessor as p
        p.clean('Preprocessor is #awesome 👍 https://github.com/s/preprocessor')

Out[5]: 'Preprocessor is'
```

Figure 66 Load Library

```
In [6]: ### Data cleaning
        # 1. with preprocessor library, designed for clearning tweets
        # loop over 'text' feature to clean
        tweets = frame.text
        p_processed_text = []
        for tweet in tweets:
            p_processed_text.append(p.clean(tweet))
        frame["text"] = p_processed_text
        print(frame.head())

       conversation_id lang                 timestamp  \
0  1231005111071100929   en  2020-02-21T23:57:44.000Z
1  1231005062165458945   en  2020-02-21T23:57:32.000Z
2  1231004398811275264   en  2020-02-21T23:54:54.000Z
3  1230866203947798530   en  2020-02-21T23:53:24.000Z
4  1231003954806501376   en  2020-02-21T23:53:08.000Z

                                              origin               author  rt  \
0  b'Italy \\U0001f1ee\\U0001f1f9 reports first l...           2985110557   2
1  b'#Coronavirus quarantine in a Hubu Village, S...  1202960464549838848   1
2  b'Italy. I love this country. #coronavirusital...  1221955502340558848   0
3  b'@JoannaB36464141 @nicolamlow @alexvespi @Ada...  1093695069176000512   0
4  b'@WHO \\nThis is a #pandemic \\n#CoronavirusO...   723319704710844417   0

   fav                                                text
0    1  b'Italy \\U0001f1ee\\U0001f1f9 reports first l...
1    2  b' quarantine in a Hubu Village, Shuangpu Town...
2    3                    b'Italy. I love this country. '
3    2  b' dispersion globallyn\\nAnalysis , travel fr...
4    0                   b' \\nThis is a \\n \\n \\n\\n '
```

Figure 67 Data Cleaning using preprocessor library

I downloaded the stopword package and remove the stop word as Figure 68, Figure 69 and Figure 70

```
In [7]: # Download stop word library

from nltk.corpus import stopwords
import nltk, os, re, string
nltk.download('stopwords')
stop = set(stopwords.words('english'))
punctuation = list(string.punctuation)
stop.update(punctuation)
```

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

Figure 68 Download the stop word

```
In [8]: # Remove Stop word

def remove_stopwords(text):
    final_text = []
    for i in text.split():
        if i.strip().lower() not in stop:
            final_text.append(i.strip())
    return " ".join(final_text)

frame['text']=frame['text'].apply(remove_stopwords)
```

Figure 69 Remove Stop word

```
In [9]: frame.head()
```

Out[9]:

	conversation_id	lang	timestamp	origin	author	rt	fav	text
0	1231005111071100929	en	2020-02-21T23:57:44.000Z	b'Italy \\U0001f1ee\\U0001f1f9 reports first l...	2985110557	2	1	b'Italy \\U0001f1ee\\U0001f1f9 reports first l...
1	1231005062165458945	en	2020-02-21T23:57:32.000Z	b'#Coronavirus quarantine in a Hubu Village, S...	1202960464549838848	1	2	b' quarantine Hubu Village, Shuangpu Town, Xih...
2	1231004398811275264	en	2020-02-21T23:54:54.000Z	b'Italy. I love this country. #coronavirusital...	1221955502340558848	0	3	b'Italy. love country.
3	1230866203947798530	en	2020-02-21T23:53:24.000Z	b'@JoannaB36464141 @nicolamlow @alexvespi @Ada...	1093695069176000512	0	2	b' dispersion globallyn\\nAnalysis travel Wuha...
4	1231003954806501376	en	2020-02-21T23:53:08.000Z	b'@WHO \\nThis is a #pandemic \\n#CoronavirusO...	723319704710844417	0	0	b' \\nThis \\n \\n \\n\\n

Figure 70 Print the data frame

I used regex to further clean the data as Figure 71 Figure 72 and remove the white space as Figure 73

```
In [10]: def cleanTweet(txt):
             txt = re.sub(r'#','',txt)
             txt = re.sub(r'RT : ','',txt)
             txt = re.sub(r'\\n','',txt)
             # to remove emojis
             txt = re.sub(r'(\s)\\U\w+', r'\1', txt)
             txt = re.sub(r'(\\u[0-9A-Fa-f]{4})', lambda matchobj: chr(int(matchobj.group(0)[2:], 16)), txt)
             txt = re.sub(emoji.get_emoji_regexp(), r"", txt)
             txt = re.sub("[^A-Za-z0-9.:,'/?!#@']", " ", txt)
             txt = re.sub(r'https?:\/\/[A-Za-z0-9\.\/]+','',txt)
             txt = re.sub(r"https?://\S+|www\.\S+","",txt)
             txt = re.sub(r"<.*?>","",txt)
             txt = re.sub(r'.', '', txt, count = 1)
             #remove the unicode starting with U000
             txt = re.sub(r'[Uu][0][0][0]\w+', '', txt)
             #remove all the signs
             txt = re.sub(r'[^\w]', ' ', txt)
             #remove all numbers
             txt = re.sub(" \d+", " ", txt)
             txt = re.sub(r'(\s)[su](\s)', ' ', txt)

             return txt

         frame["text"] = frame["text"].apply(cleanTweet)
```

Figure 71 Further data clean

```
In [11]: frame.head()
```

Out[11]:

	conversation_id	lang	timestamp	origin	author	rt	fav	text
0	1231005111071100929	en	2020-02-21T23:57:44.000Z	b'Italy \\U0001f1ee\\U0001f1f9 reports first l...	2985110557	2	1	Italy reports first locally transmitted ...
1	1231005062165458945	en	2020-02-21T23:57:32.000Z	b'#Coronavirus quarantine in a Hubu Village, S...	1202960464549838848	1	2	quarantine Hubu Village Shuangpu Town Xihu...
2	1231004398811275264	en	2020-02-21T23:54:54.000Z	b'Italy. I love this country. #coronavirusital...	1221955502340558848	0	3	Italy love country
3	1230866203947798530	en	2020-02-21T23:53:24.000Z	b'@JoannaB36464141 @nicolamlow @alexvespi @Ada...	1093695069176000512	0	2	dispersion globallyn Analysis travel Wuhan ...
4	1231003954806501376	en	2020-02-21T23:53:08.000Z	b'@WHO \\nThis is a #pandemic \\n#CoronavirusO...	723319704710844417	0	0	This

Figure 72 Recheck Data

```
In [12]: #remove leading/trailing whitespaces on the text column
         frame['text'] = frame['text'].str.strip()

In [13]: nan_value = float("NaN")
         frame.replace("", nan_value, inplace=True)
         frame.dropna(subset = ["text"], inplace=True)
```

Figure 73 Remove white space

I generated the sentiment score as Figure 74 Figure 75

```
In [14]: from textblob import TextBlob
         def sentiment_text(text):
             sent_sentences=[]
             blob = TextBlob(text)
             for sentence in blob.sentences:
                 sent_sentences.append(sentence.sentiment.polarity)
             return sum(sent_sentences)/float(len(sent_sentences))
```

```
In [15]: frame["sentiment"] = frame["text"].apply(sentiment_text)
```

Figure 74 Generate the sentiment score

```
In [16]: frame.head()
Out[16]:
```

	conversation_id	lang	timestamp	origin	author	rt	fav	text	sentiment
0	1231005111071100929	en	2020-02-21T23:57:44.000Z	b'Italy \\U0001f1ee\\U0001f1f9 reports first l...	2985110557	2	1	Italy reports first locally transmitted c...	0.083333
1	1231005062165458945	en	2020-02-21T23:57:32.000Z	b'#Coronavirus quarantine in a Hubu Village, S...	1202960464549838848	1	2	quarantine Hubu Village Shuangpu Town Xihu D...	0.000000
2	1231004398811275264	en	2020-02-21T23:54:54.000Z	b'Italy. I love this country. #coronavirusital...	1221955502340558848	0	3	Italy love country	0.500000
3	1230866203947798530	en	2020-02-21T23:53:24.000Z	b'@JoannaB36464141 @nicolamlow @alexvespi @Ada...	1093695069176000512	0	2	dispersion globallyn Analysis travel Wuhan av...	0.400000
4	1231003954806501376	en	2020-02-21T23:53:08.000Z	b'@WHO \\nThis is a #pandemic \\n#CoronavirusO...	723319704710844417	0	0	This	0.000000

Figure 75 Recheck date

I lemmatized text as Figure 76 and rename column in Figure 77, Figure 78, Figure 79, Figure 80 before store the dataframe in Figure 81

```
In [17]: tweets = frame["text"]
         tokenized_tweet = []
         for tweet in tweets:
             tokenized_tweet.append(nltk.word_tokenize(tweet))
         frame["Tokenized_Tweet"] = tokenized_tweet
```

```
In [18]: # Lemmatization
         wordnet_lemmatizer = WordNetLemmatizer()
         lemmatized_text = []

         for index, row in frame.iterrows():
             lemma_article = []
             row = row['Tokenized_Tweet']
             for w in row:
                 word1 = wordnet_lemmatizer.lemmatize(w, pos = "n")
                 word2 = wordnet_lemmatizer.lemmatize(word1, pos = "v")
                 word3 = wordnet_lemmatizer.lemmatize(word2, pos = ("a"))
                 lemma_article.append(word3)
             lemmatized_text.append(lemma_article)
         print(lemmatized_text)
         frame['lemmatized_text'] = lemmatized_text
```

Figure 76 Lemmatize Text

```
In [21]: frame[["conversation_id","text","Tokenized_Tweet","lemmatized_text"]].head()
```

Out[21]:

	conversation_id	text	Tokenized_Tweet	lemmatized_text
0	1231005111071100929	Italy reports first locally transmitted c...	[Italy, reports, first, locally, transmitted, ...	[Italy, report, first, locally, transmit, case...
1	1231005062165458945	quarantine Hubu Village Shuangpu Town Xihu D...	[quarantine, Hubu, Village, Shuangpu, Town, Xi...	[quarantine, Hubu, Village, Shuangpu, Town, Xi...
2	1231004398811275264	Italy love country	[Italy, love, country]	[Italy, love, country]
3	1230866203947798530	dispersion globallyn Analysis travel Wuhan av...	[dispersion, globallyn, Analysis, travel, Wuha...	[dispersion, globallyn, Analysis, travel, Wuha...
4	1231003954806501376	This	[This]	[This]

```
In [28]: # Rename column
         frame.rename(columns = {'conversation_id':'id'}, inplace = True)
         frame.rename(columns = {'created_at':'timestamp'}, inplace = True)
```

Figure 77 Rename the column name

```
In [29]: frame.describe(include=['number'])
```

Out[29]:

	id	author	rt	fav	sentiment
count	1.628414e+06	1.628414e+06	1.628414e+06	1.628414e+06	1.628414e+06
mean	1.252350e+18	3.599463e+17	2.600770e+00	8.035027e+00	8.990087e-02
std	1.315489e+16	5.086149e+17	1.541226e+02	3.188259e+02	2.442936e-01
min	1.837335e+09	5.090000e+02	0.000000e+00	0.000000e+00	-1.000000e+00
25%	1.243973e+18	1.594716e+08	0.000000e+00	0.000000e+00	0.000000e+00
50%	1.251208e+18	1.586448e+09	0.000000e+00	0.000000e+00	0.000000e+00
75%	1.260616e+18	9.167542e+17	1.000000e+00	2.000000e+00	2.000000e-01
max	1.274855e+18	1.274780e+18	1.404620e+05	1.896340e+05	1.000000e+00

Figure 78 Recheck Data

```
In [32]: print(frame.dtypes)

id                   int64
lang                object
timestamp           object
origin              object
author               int64
rt                   int64
fav                  int64
text                object
sentiment          float64
Tokenized_Tweet     object
lemmatized_text     object
dtype: object
```

Figure 79 Recheck Column

```
In [33]: print(frame.shape)

(1628414, 11)
```

Figure 80 Check the dataframe size

```
In [34]: frame.to_csv(r'C:\Users\weihe\Documents\TweetData\tweet_dataframe_process.csv', index = False, h
```

Figure 81 Save Dataframe

4.2.2 Reddit Data

To pre-process the reddit data, I have performed the similar procedure to as twitter data as Figure 82 to Figure 96

```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns

In [2]: frame = pd.read_csv("reddit_aggregated.csv")

In [3]: frame.head()
```

Out[3]:

	id	origin	Score	timestamp	comments	text
0	f2kpf6	Found something in "The Lancet" regarding cyto...	1	2020-02-12 03:28:58	0	Found something in "The Lancet" regarding cyto...
1	f2m2cv	Any info on how COVID-19 affect pets/domestic ...	1	2020-02-12 05:13:07	5	Any info on how COVID-19 affect pets/domestic ...
2	f2p51b	Johns Hopkins Bloomberg school of public healt...	1	2020-02-12 10:16:11	0	Johns Hopkins Bloomberg school of public healt...
3	f2qh4a	Charge for Victory!!! Smash down the COVID vir...	1	2020-02-12 12:28:14	0	Charge for Victory!!! Smash down the COVID vir...
4	f2roo5	China Reports Smallest Number Of New COVID-19 ...	1	2020-02-12 14:04:57	5	China Reports Smallest Number Of New COVID-19 ...

```
In [4]: frame.shape
Out[4]: (10652, 6)
```

Figure 82 Load libray and data

```
In [5]: import preprocessor as p

In [6]: ### Data cleaning
        # 1. with preprocessor library, deisnged for clearning tweets
        # loop over 'text' feature to clean
        reddits = frame.text
        p_processed_text = []
        for reddit in reddits:
            p_processed_text.append(p.clean(reddit))
        frame["text"] = p_processed_text
        print(frame.head())

       id                                             origin  Score  \
0  f2kpf6  Found something in "The Lancet" regarding cyto...      1
1  f2m2cv  Any info on how COVID-19 affect pets/domestic ...      1
2  f2p51b  Johns Hopkins Bloomberg school of public healt...      1
3  f2qh4a  Charge for Victory!!! Smash down the COVID vir...      1
4  f2roo5  China Reports Smallest Number Of New COVID-19 ...      1

             timestamp  comments  \
0  2020-02-12 03:28:58         0
1  2020-02-12 05:13:07         5
2  2020-02-12 10:16:11         0
3  2020-02-12 12:28:14         0
4  2020-02-12 14:04:57         5

                                                text
0  Found something in "The Lancet" regarding cyto...
1  Any info on how COVID-19 affect pets/domestic ...
2  Johns Hopkins Bloomberg school of public healt...
3  Charge for Victory!!! Smash down the COVID vir...
4  China Reports Smallest Number Of New COVID-19 ...
```

Figure 83 Apply the preprocessor library

```
In [7]: from nltk.corpus import stopwords
        import nltk, os, re, string
        nltk.download('stopwords')
        stop = set(stopwords.words('english'))
        punctuation = list(string.punctuation)
        stop.update(punctuation)

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

Figure 84 Download stopwords library

```
In [8]: def remove_stopwords(text):
            final_text = []
            for i in text.split():
                if i.strip().lower() not in stop:
                    final_text.append(i.strip())
            return " ".join(final_text)

        frame['text']=frame['text'].apply(remove_stopwords)
```

Figure 85 Define the remove stopwords procedure

```
In [9]: frame.head()
Out[9]:
```

	id	origin	Score	timestamp	comments	text
0	f2kpf6	Found something in "The Lancet" regarding cyto...	1	2020-02-12 03:28:58	0	Found something "The Lancet" regarding cytokin...
1	f2m2cv	Any info on how COVID-19 affect pets/domestic ...	1	2020-02-12 05:13:07	5	info COVID-19 affect pets/domestic animals?
2	f2p51b	Johns Hopkins Bloomberg school of public healt...	1	2020-02-12 10:16:11	0	Johns Hopkins Bloomberg school public health C...
3	f2qh4a	Charge for Victory!!! Smash down the COVID vir...	1	2020-02-12 12:28:14	0	Charge Victory!!! Smash COVID virus!!! Wipe ba...
4	f2roo5	China Reports Smallest Number Of New COVID-19 ...	1	2020-02-12 14:04:57	5	China Reports Smallest Number New COVID-19 Cas...

Figure 86 Check the data

```
In [10]: import nltk
         from nltk.stem import WordNetLemmatizer
         import re
         from nltk.corpus import stopwords
         nltk.download('punkt')
         nltk.download('wordnet')
         lemma = WordNetLemmatizer()
         def process_text(text):
             text = re.sub("(@[A-Za-z0-9_]+)|([^0-9A-Za-z \t])", " ",text.lower())
             words = nltk.word_tokenize(text)
             words = [lemma.lemmatize(word) for word in words if word not in set(stopwords.words("english"))]
             text = " ".join(words)

             return text

         frame["text"] = frame["text"].apply(process_text)

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

Figure 87 process the data

```
In [11]: frame.head()
Out[11]:
```

	id	origin	Score	timestamp	comments	text
0	f2kpf6	Found something in "The Lancet" regarding cyto...	1	2020-02-12 03:28:58	0	found something lancet regarding cytokine stor...
1	f2m2cv	Any info on how COVID-19 affect pets/domestic ...	1	2020-02-12 05:13:07	5	info covid 19 affect pet domestic animal
2	f2p51b	Johns Hopkins Bloomberg school of public healt...	1	2020-02-12 10:16:11	0	john hopkins bloomberg school public health co...
3	f2qh4a	Charge for Victory!!! Smash down the COVID vir...	1	2020-02-12 12:28:14	0	charge victory smash covid virus wipe bad viru...
4	f2roo5	China Reports Smallest Number Of New COVID-19 ...	1	2020-02-12 14:04:57	5	china report smallest number new covid 19 case...

Figure 88 Check Data

```
In [12]: import emoji

         def cleanReddit(txt):
             txt = re.sub(r'#','',txt)
             txt = re.sub(r'RT : ','',txt)
             txt = re.sub(r'\n','',txt)
             # to remove emojis
             txt = re.sub(emoji.get_emoji_regexp(), r"", txt)
             txt = re.sub(r'https?:\/\/[A-Za-z0-9\.\/]+','',txt)
             txt = re.sub(r"https?://\S+|www\.\S+","",txt)
             txt = re.sub(r"<.*?>","",txt)
             return txt

         frame["text"] = frame["text"].apply(cleanReddit)
```

Figure 89 Further clean data

```
In [13]: frame.head()
```

Out[13]:

	id	origin	Score	timestamp	comments	text
0	f2kpf6	Found something in "The Lancet" regarding cyto...	1	2020-02-12 03:28:58	0	found something lancet regarding cytokine stor...
1	f2m2cv	Any info on how COVID-19 affect pets/domestic ...	1	2020-02-12 05:13:07	5	info covid 19 affect pet domestic animal
2	f2p51b	Johns Hopkins Bloomberg school of public healt...	1	2020-02-12 10:16:11	0	john hopkins bloomberg school public health co...
3	f2qh4a	Charge for Victory!!! Smash down the COVID vir...	1	2020-02-12 12:28:14	0	charge victory smash covid virus wipe bad viru...
4	f2roo5	China Reports Smallest Number Of New COVID-19 ...	1	2020-02-12 14:04:57	5	china report smallest number new covid 19 case...

Figure 90 Check the data

```
In [15]: frame[frame['text'] == ''].index
Out[15]: Int64Index([196, 1708, 4432, 4533, 6777, 6820, 7357, 7435, 9158, 9271, 10352], dtype='int64')

In [16]: nan_value = float("NaN")
         frame.replace("", nan_value, inplace=True)
         frame.dropna(subset = ["text"], inplace=True)
```

Figure 91 Remove the null data rows

```
In [17]: from textblob import TextBlob
         def sentiment_text(text):

             sent_sentences=[]
             blob = TextBlob(text)
             for sentence in blob.sentences:
                 sent_sentences.append(sentence.sentiment.polarity)
             return sum(sent_sentences)/float(len(sent_sentences))

In [18]: frame["sentiment"] = frame["text"].apply(sentiment_text)
```

Figure 92 Calculate the sentiment

```
In [19]: reddits = frame["text"]
         tokenized_reddit = []
         for reddit in reddits:
             tokenized_reddit.append(nltk.word_tokenize(reddit))
         frame["Tokenized_Reddit"] = tokenized_reddit
```

```
In [20]: frame.head()
```

Out[20]:

	id	origin	Score	timestamp	comments	text	sentiment	Tokenized_Reddit
0	f2kpf6	Found something in "The Lancet" regarding cyto...	1	2020-02-12 03:28:58	0	found something lancet regarding cytokine stor...	0.000000	[found, something, lancet, regarding, cytokine...
1	f2m2cv	Any info on how COVID-19 affect pets/domestic ...	1	2020-02-12 05:13:07	5	info covid 19 affect pet domestic animal	0.000000	[info, covid, 19, affect, pet, domestic, animal]
2	f2p51b	Johns Hopkins Bloomberg school of public healt...	1	2020-02-12 10:16:11	0	john hopkins bloomberg school public health co...	0.000000	[john, hopkins, bloomberg, school, public, hea...
3	f2qh4a	Charge for Victory!!! Smash down the COVID vir...	1	2020-02-12 12:28:14	0	charge victory smash covid virus wipe bad viru...	-0.700000	[charge, victory, smash, covid, virus, wipe, b...
4	f2roo5	China Reports Smallest Number Of New COVID-19 ...	1	2020-02-12 14:04:57	5	china report smallest number new covid 19 case...	-0.088068	[china, report, smallest, number, new, covid, ...

Figure 93 Tokenize the text

```
In [21]: # Lemmatization
         wordnet_lemmatizer = WordNetLemmatizer()
         lemmatized_text = []

         for index, row in frame.iterrows():
             lemma_article = []
             row = row['Tokenized_Reddit']
             for w in row:
                 word1 = wordnet_lemmatizer.lemmatize(w, pos = "n")
                 word2 = wordnet_lemmatizer.lemmatize(word1, pos = "v")
                 word3 = wordnet_lemmatizer.lemmatize(word2, pos = ("a"))
                 lemma_article.append(word3)
             lemmatized_text.append(lemma_article)
         print(lemmatized_text)
         frame['lemmatized_text'] = lemmatized_text
```

```
on', 'covid', '19'], ['covid'], ['update', 'covid', '19', 'outbreak', 'professor', 'neil', 'ferguson', 'dr', 'ilaria', 'dorig
atti', 'dr', 'lucy', 'okell'], ['case', 'index', 'patient', 'cause', 'tertiary', 'transmission', 'coronavirus', 'disease', 'k
orea', 'application', 'lopinavir', 'ritonavir', 'treatment', 'covid', '19', 'pneumonia', 'monitor', 'quantitative', 'rt', 'pc
r', 'korean', 'academy', 'medical', 'science', 'publish', 'online', 'feb', 'academic', 'report'], ['aspirin', 'make', 'covi
d', '19', 'bad', 'salicylate', 'pandemic', 'influenza', 'mortality', 'pharmacology', 'pathology', 'historic', 'evidence', 'cl
inical', 'infectious', 'disease'], ['im', 'hard', 'time', 'understand', 'u', 'response', 'time', 'covid', '19', 'threat', 'pe
rhaps', 'help', 'clarify'], ['analysis', 'covid', '19', 'north', 'korea'], ['analysis', 'covid', '19', 'north', 'korea'], ['c
irculatory', 'appearance', 'lung', 'infect', 'corona', 'covid', '19', 'virus'], ['circulatory', 'appearance', 'lung', 'infec
t', 'corona', 'covid', '19', 'virus'], ['fact', 'check', 'request', 'lancet', 'medical', 'journal', 'publish', 'paper', 'edit
orial', 'assert', 'covid', '19', 'show', 'evidence', 'genetically', 'modify', 'bioweapon'], ['san', 'diego', 'lab', 'discove
r', 'covid', '19', 'vaccine', 'hour'], ['china', 'cdc', 'weekly', 'epidemiological', 'characteristic', 'outbreak', 'novel',
'coronavirus', 'disease', 'covid', '19', 'china'], ['calculate', 'covid', '19', 'requirement', 'overwhelm', 'u', 'healthcar
e', 'system'], ['lancet', '17', '20', 'pathological', 'find', 'covid', '19', 'associate', 'acute', 'respiratory', 'distress',
'syndrome'], ['chest', 'ct', 'image', 'covid', '19', 'lung', 'involvement', 'year', 'old', 'huanan', 'seafood', 'worker', 'da
y', 'symptom', 'progression', 'die', 'day', 'late'], ['n95', 'mask', 'effective', 'particle', 'large', 'small', 'covid', '1
9', '0', '12', 'micron', 'minimum', 'effectiveness'], ['lancet', '17', '20', 'pathological', 'find', 'covid', '19', 'associat
e', 'acute', 'respiratory', 'distress', 'syndrome'], ['fatality', 'rate', 'covid', '19', 'different', 'age', 'group'], ['fata
lity', 'rate', 'resolve', 'case', 'covid', '19', 'different', 'age', 'group'], ['resolve', 'cfr', 'covid', '19', 'different',
'age', 'group'], ['pathological', 'find', 'covid', '19', 'associate', 'acute', 'respiratory', 'distress', 'syndrome'], ['aa
a', 'covid', '19', 'panel', 'include', 'bill', 'melinda', 'gate', 'foundation', 'discus', 'severity', 'outbreak', 'track', 'v
```

Figure 94 Lemmatize the text

```
In [22]: frame.head()
```

Out[22]:

	id	origin	Score	timestamp	comments	text	sentiment	Tokenized_Reddit	lemmatized_text
0	f2kpf6	Found something in "The Lancet" regarding cyto...	1	2020-02-12 03:28:58	0	found something lancet regarding cytokine stor...	0.000000	[found, something, lancet, regarding, cytokine...	[find, something, lancet, regard, cytokine, st...
1	f2m2cv	Any info on how COVID-19 affect pets/domestic ...	1	2020-02-12 05:13:07	5	info covid 19 affect pet domestic animal	0.000000	[info, covid, 19, affect, pet, domestic, animal]	[info, covid, 19, affect, pet, domestic, animal]
2	f2p51b	Johns Hopkins Bloomberg school of public healt...	1	2020-02-12 10:16:11	0	john hopkins bloomberg school public health co...	0.000000	[john, hopkins, bloomberg, school, public, hea...	[john, hopkins, bloomberg, school, public, hea...
3	f2qh4a	Charge for Victory!!! Smash down the COVID vir...	1	2020-02-12 12:28:14	0	charge victory smash covid virus wipe bad viru...	-0.700000	[charge, victory, smash, covid, virus, wipe, b...	[charge, victory, smash, covid, virus, wipe, b...
4	f2roo5	China Reports Smallest Number Of New COVID-19 ...	1	2020-02-12 14:04:57	5	china report smallest number new covid 19 case...	-0.088068	[china, report, smallest, number, new, covid, ...	[china, report, small, number, new, covid, 19,...

Figure 95 Check the data

```
In [24]: frame.to_csv(r'C:\Users\weihe\Documents\RedditData\export_reddit_dataset.csv', index = False,
```

Figure 96 Store the data

4.3 Exploration Analysis

Exploration analysis is done using Python to find the patter in the test feature. I analysed the dataset from Twitter and Reddit.

4.3.1 Tweet Data exploration

I loaded the tweets data and extract the lemmatized_text field as Figure 97 and Figure 98

```
In [1]: import numpy as np
        from PIL import Image
        import pandas as pd
        from wordcloud import WordCloud, ImageColorGenerator, STOPWORDS
        import matplotlib.pyplot as plt
```

Figure 97 Load library

```
In [2]: df1 = pd.read_csv('tweet_dataframe_process2.csv')

In [3]: lem = df1['lemmatized_text'].values.tolist()

In [5]: text = " ".join(review for review in df1.lemmatized_text.astype(str))
```

Figure 98 load data and join the lematized text

```
In [6]: print ("There are {} words in the combination of all cells in column YOUR_COLUMN_NAME.".format(len(text)))
        There are 11837820 words in the combination of all cells in column YOUR_COLUMN_NAME.
```

Figure 99 Print statistics

I ran the wordcloud model to create the word count as Figure 100

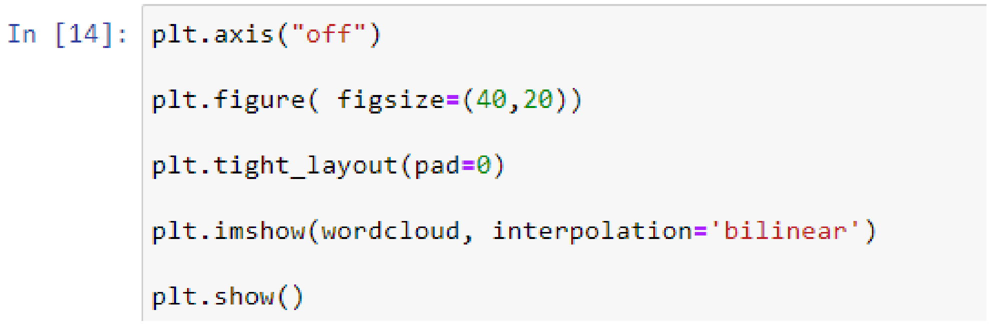
```
In [7]: stopwords = set(STOPWORDS)

In [11]: WordCloud().generate(text)
Out[11]: <wordcloud.wordcloud.WordCloud at 0x2954976c7f0>

In [12]: wordcloud = WordCloud(stopwords=stopwords, background_color="white", width=800, height=400).generate(text)
```

Figure 100 Compute the model for wordcloud

I printed out the plot generate by wordcloud as Figure 101 and Figure 102



```
In [14]: plt.axis("off")

         plt.figure( figsize=(40,20))

         plt.tight_layout(pad=0)

         plt.imshow(wordcloud, interpolation='bilinear')

         plt.show()
```

Figure 101 Design plot the wordcloud model

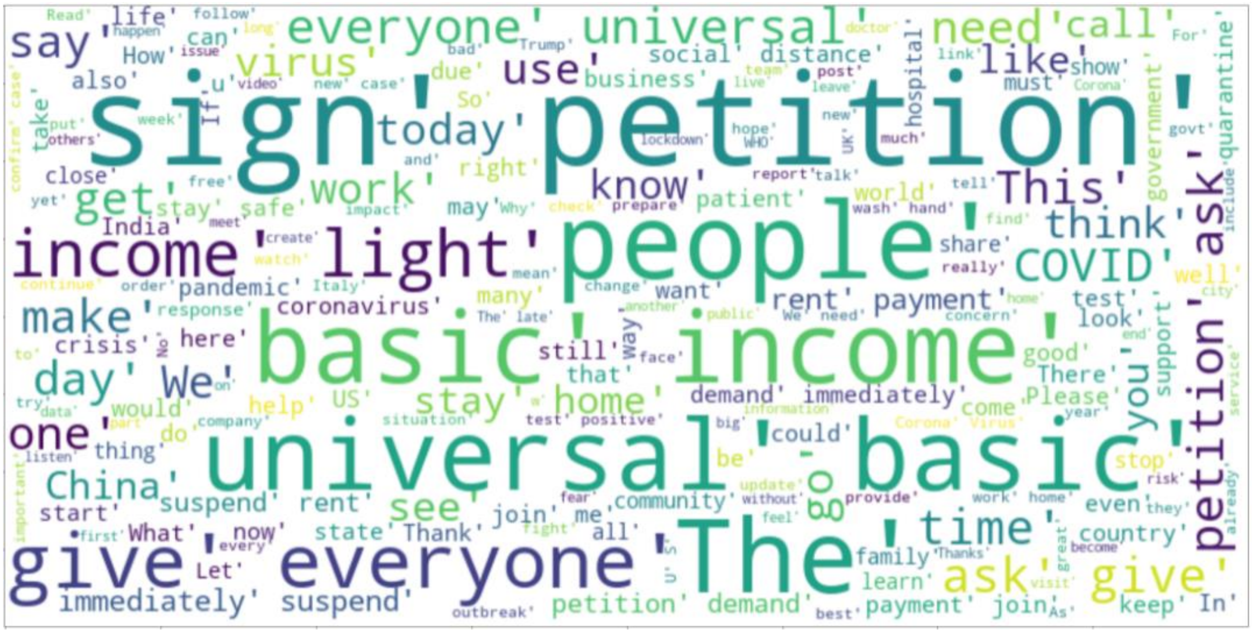
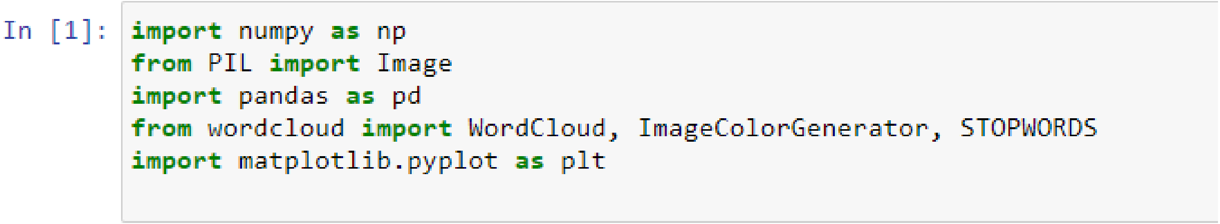


Figure 102 Display the model output

4.3.2 Reddit exploration

I have repeated the same procedure on the Reddit as Figure 103 to Figure 107.



```
In [1]: import numpy as np
        from PIL import Image
        import pandas as pd
        from wordcloud import WordCloud, ImageColorGenerator, STOPWORDS
        import matplotlib.pyplot as plt
```

Figure 103 Load library

```
In [2]: df1 = pd.read_csv('export_reddit_dataset.csv')

In [3]: lem = df1['lemmatized_text'].values.tolist()

In [4]: text = " ".join(review for review in df1.lemmatized_text.astype(str))

In [5]: print ("There are {} words in the combination of all cells in column YOUR_COLUMN_NAME.".format(len(text)))
        There are 952532 words in the combination of all cells in column YOUR_COLUMN_NAME.
```

Figure 104 Load the data and join the lemmatized text

```
In [6]: stopwords = set(STOPWORDS)

In [7]: WordCloud().generate(text)
Out[7]: <wordcloud.wordcloud.WordCloud at 0x1ec3e06ca30>

In [8]: wordcloud = WordCloud(stopwords=stopwords, background_color="white", width=800, height=400).generate(text)
```

Figure 105 Generate the WordCloud

```
In [9]: plt.axis("off")
        plt.figure( figsize=(40,20))
        plt.tight_layout(pad=0)
        plt.imshow(wordcloud, interpolation='bilinear')
        plt.show()
```

Figure 106 Create the plot for the model

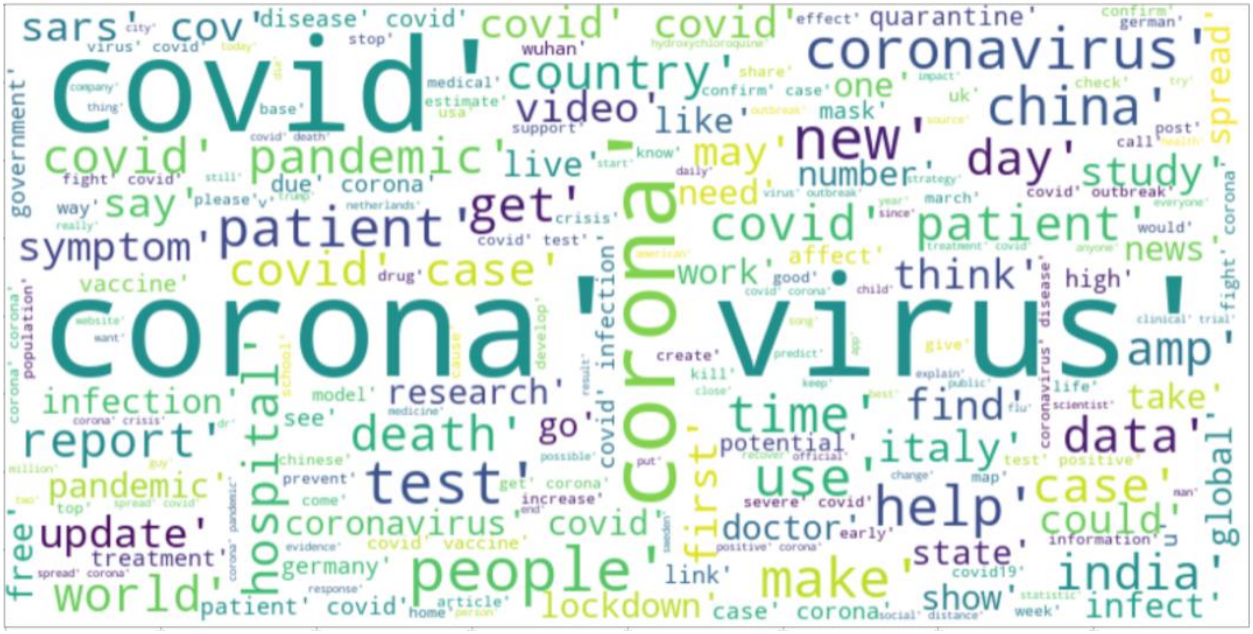


Figure 107 Display the model

5 Implementation

I used Jupyter notebook to implement the LDA topic modelling and sentiment analysis

5.1 Topic Modelling

5.1.1 Twitter Data

I started by load the library as Figure 109 extract the lemmatized text as Figure 110.

```
In [1]: import gensim
        import gensim.corpora as corpora
        from gensim.corpora import Dictionary
        from gensim.models.coherencemodel import CoherenceModel
        from gensim.models.ldamodel import LdaModel

        from pprint import pprint

        import spacy

        import pickle
        import re
        import pyLDAvis
        import pyLDAvis.gensim_models

        import matplotlib.pyplot as plt
        import pandas as pd

        C:\Users\weihe\Anaconda3\lib\site-packages\gensim\similarities\__init__.py:15: Use
        evenshtein package <https://pypi.org/project/python-Levenshtein/> is unavailable.
          warnings.warn(msg)
```

Figure 108 Load library

```
In [2]: df = pd.read_csv('tweet_dataframe_process.csv')
        df.head()
```

Out[2]:

	id	lang	timestamp	origin	author	rt	fav	text	sentiment	Tokenized_Tweet
0	1231005111071100929	en	2020-02-21T23:57:44.000Z	b'Italy \\U0001f1ee\\U0001f1f9 reports first l...	2985110557	2	1	Italy reports first locally transmitted c...	0.083333	['Italy', 'reports', 'first', 'locally', 'tran... [
1	1231005062165458945	en	2020-02-21T23:57:32.000Z	b'#Coronavirus quarantine in a Hubu Village, S...	1202960464549838848	1	2	quarantine Hubu Village Shuangpu Town Xihu D...	0.000000	['quarantine', 'Hubu', 'Village', 'Shuangpu', ... [''
2	1231004398811275264	en	2020-02-21T23:54:54.000Z	b'Italy. I love this country. #coronavirusital...	1221955502340558848	0	3	Italy love country	0.500000	['Italy', 'love', 'country']
3	1230866203947798530	en	2020-02-21T23:53:24.000Z	b'@JoannaB36464141 @nicolamlow @alexvespi @Ada...	1093695069176000512	0	2	dispersion globallyn Analysis travel Wuhan av...	0.400000	['dispersion', 'globallyn', 'Analysis', 'trave...
4	1231003954806501376	en	2020-02-21T23:53:08.000Z	b'@WHO \\nThis is a #pandemic \\n#CoronavirusO...	723319704710844417	0	0	This	0.000000	['This']

Figure 109 Load CSV data

```
In [3]: lem = df['lemmatized_text'].values.tolist()

        lemmatized_list = []
        for lemmatized_item in lem:
            print(lemmatized_item)
            lemmatized_list.append(eval(lemmatized_item))

        print(lemmatized_list[:2])

['Italy', 'report', 'first', 'locally', 'transmit', 'case', 'coronavirus', 'Iran', 'death', 'toll', 'rise', 'four', 'ABC', 'News', 'Australian', 'Corporat
['quarantine', 'Hubu', 'Village', 'Shuangpu', 'Town', 'Xihu', 'District', 'China', 'Chinese', 'authority', 'deploy', 'giant', 'fog', 'machine', 'even', 's
eo', 'Onebtcer']
['Italy', 'love', 'country']
['dispersion', 'globallyn', 'Analysis', 'travel', 'Wuhan', 'available', 'Jan', 'travel', 'China', 'available', 'Jan', 'ha', 'Teheran', 'destination']
['This']
['warn', 'community', 'spread', 'could', 'take', 'place', 'US']
['After', 'thorough', 'investigation', 'time', 'evidence', 'patient', 'expose', 'visit']
['It', 'look', 'like', 'force', 'removal', 'isolation', 'quarantine', 'CCP', 'act', 'erratically']
['CDC', 'didn', 't', 'want', 'patient', 'fly', 'US', 'it', 'overrule']
['N100', 'Disposable', 'Respirator', 'Cool', 'Flow', 'Exhalation', 'Valve', 'CASE']
['Group', 'Hongkongers', 'set', 'surgical', 'face', 'mask', 'factory', 'overnight', 'ease', 'supply', 'amid', 'coronavirus', 'scare']
['M', 'Respirator', 'N95', 'Cool', 'Flow', 'Valve', 'Pack']
['I', 'put', 'commentator', 'journalist', 'hat', 'best', 'summary', 'interview', 'virologist', 'relate', 'investigation', 'inform', 'comment', 'welcome',
w', 'include', 'Warning', 'intense']
['Wondering', 'win', 'Fantasy', 'Sports', 'crush', 'competition', 'Introducing', 'new', 'Z', 'Code', 'Daily', 'Fantasy', 'Predictor']
['Amazon', 'Virus', 'Killing', 'Device']
['Feb', 'Director', 'Says', 'World', 'Must', 'Act', 'Fast', 'Contain', 'Cases', 'Dead']
['BREAKING', 'Italy', 'report', 'first', 'death', 'Earleir', 'Italian', 'official', 'order', 'mass', 'closure', 'include', 'public', 'build', 'restaurant'
```

Figure 110 Extract Lemmatized text

I computed for id2word as Figure 111 and build LDA model as Figure 112

```
In [4]: # Create the dictionary
        id2word = corpora.Dictionary(lemmatized_list)

        # Create corpus
        texts = lemmatized_list

        # Term Document Frequency
        corpus = [id2word.doc2bow(text) for text in texts]

        #View
        print(corpus[:1])

[[(0, 1), (1, 1), (2, 1), (3, 1), (4, 1), (5, 1), (6, 1), (7, 1), (8, 1), (9, 1), (10, 1), (11, 1), (12, 1), (13, 1), (14, 1), (15, 1)]]

In [5]: id2word[0]

Out[5]: 'ABC'
```

Figure 111 Compute id2word and display the first element

```
In [7]: #Build Topic model

        lda_model = gensim.models.ldamodel.LdaModel(corpus=corpus,
                                                    id2word=id2word,
                                                    num_topics=4,
                                                    random_state=100,
                                                    update_every=1,
                                                    chunksize=100,
                                                    passes=10,
                                                    alpha='auto',
                                                    per_word_topics=True)
```

Figure 112 Build the LDA model with 4 topics

Model display and coherence score can be found in the Figure 113 Figure 114 Figure 115

```
In [8]: pprint(lda_model.print_topics())
        doc_lda=lda_model[corpus]

        [(0,
          '0.026*"case" + 0.024*"COVID" + 0.017*"Coronavirus" + 0.013*"new" + '
          '0.012*"update" + 0.011*"come" + 0.009*"today" + 0.009*"social" + '
          '0.008*"number" + 0.008*"US"'),
         (1,
          '0.023*"ask" + 0.022*"sign" + 0.021*"petition" + 0.021*"income" + '
          '0.020*"basic" + 0.020*"universal" + 0.015*"need" + 0.014*"time" + '
          '0.013*"go" + 0.012*"help"'),
         (2,
          '0.033*"give" + 0.028*"light" + 0.013*"like" + 0.012*"know" + 0.010*"good" + '
          '0.010*"one" + 0.009*"close" + 0.009*"think" + 0.008*"would" + 0.008*"keep"'),
         (3,
          '0.032*"everyone" + 0.021*"people" + 0.019*"get" + 0.016*"test" + '
          '0.015*"day" + 0.015*"The" + 0.013*"u" + 0.012*"virus" + 0.012*"make" + '
          '0.011*"say"')]
```

Figure 113 Display the topic from the model.

```
In [9]: print('Perplexity: ', lda_model.log_perplexity(corpus))

        # Compute Coherence Score
        coherence_model_lda = CoherenceModel(model=lda_model, texts=lemmatized_list, dictionary=id2word, coherence='c_v')
        coherence_lda = coherence_model_lda.get_coherence()
        print('Coherence Score: ', coherence_lda)

        Perplexity:  -8.862477314736944
        Coherence Score:  0.20355419839742395
```

Figure 114 Display Perplexity and Coherence score

```
In [10]: # Visualise the topic keyword
         pyLDAvis.enable_notebook()
         vis = pyLDAvis.gensim_models.prepare(lda_model, corpus, id2word)
         vis
```

Figure 115 Display generated model

I checked the intertopic distance for all four topics below. Figure 116 to Figure 119

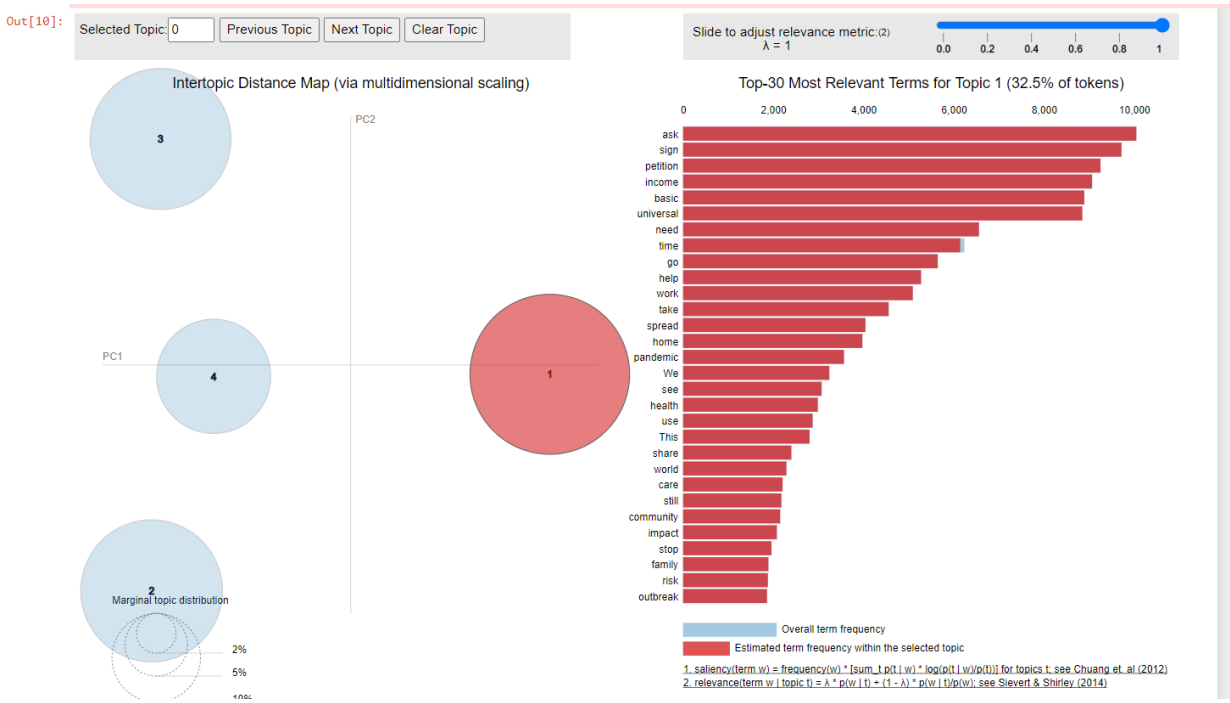


Figure 116 Display Topic 1

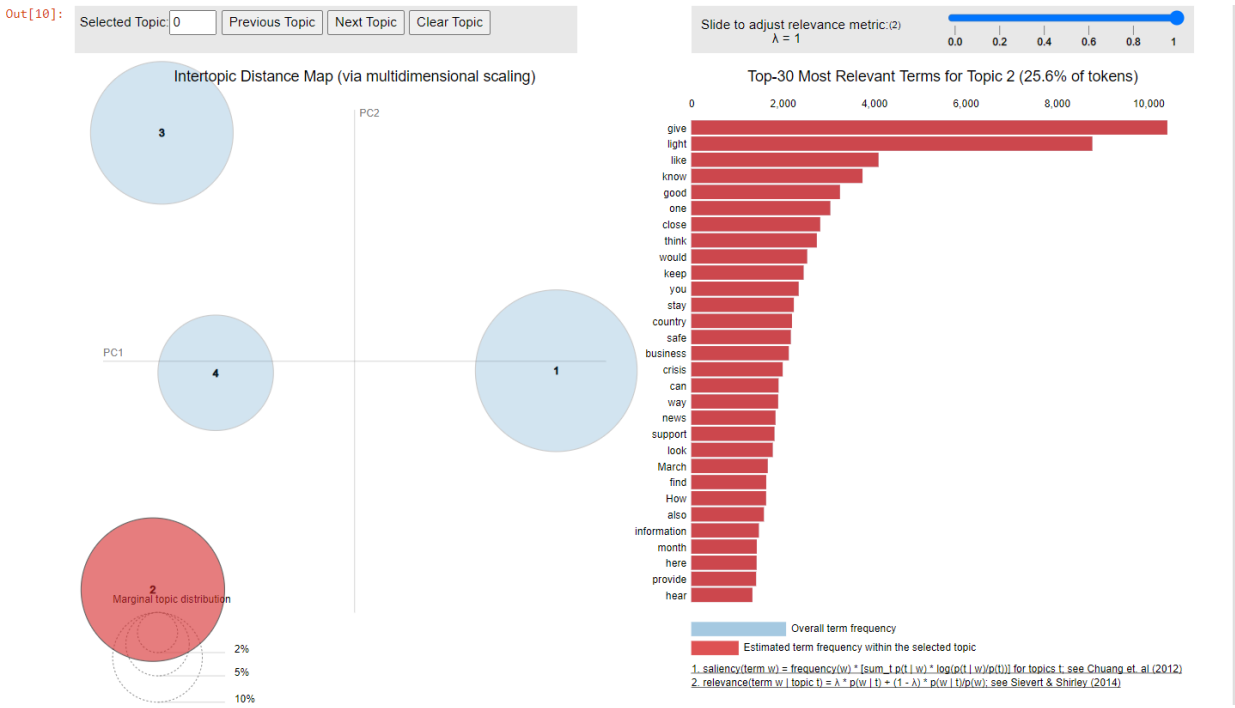


Figure 117 Display Topic 2

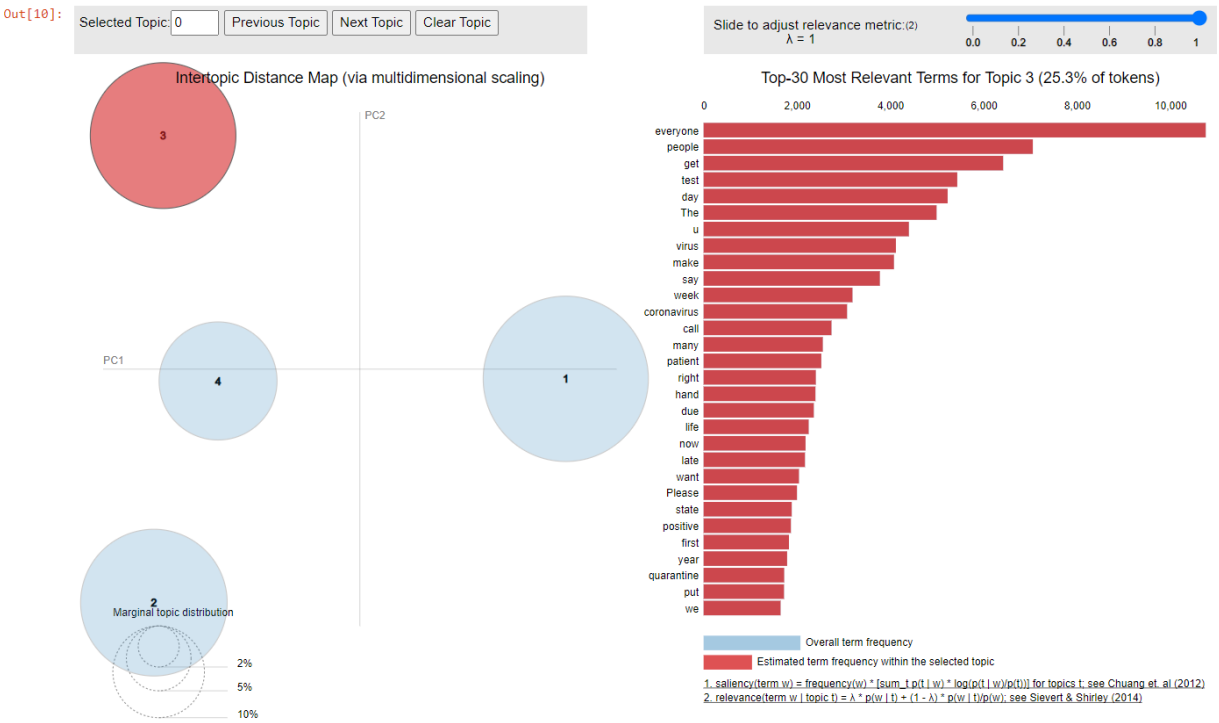


Figure 118 Display Topic 3

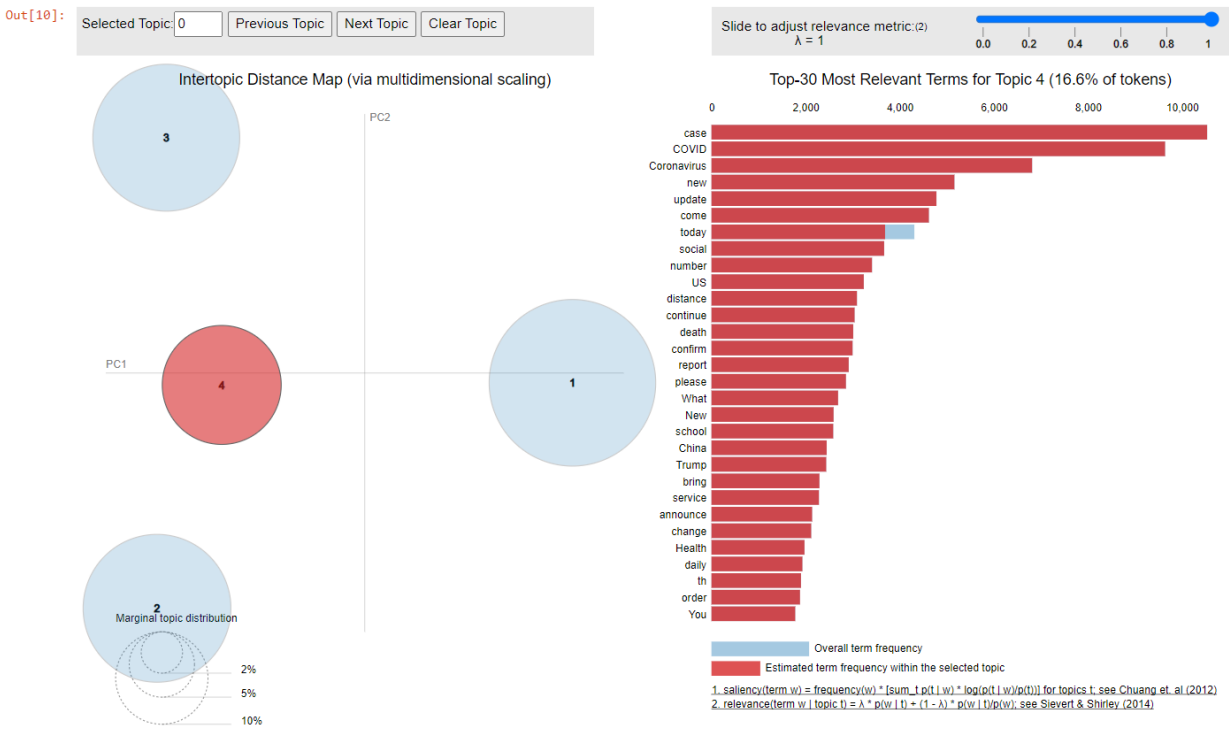


Figure 119 Display Topic 4

I loaded the Mallet library to calculate the coherence score and found the most suitable topics ad Figure 120 to Figure 126

```
In [11]: import os

         os.environ.update({'MALLET_HOME':r'C:/Users/weihe/Documents/TweetData/mallet-2.0.8/mallet-2.0.8'})
         mallet_path = r'C:/Users/weihe/Documents/TweetData/mallet-2.0.8/mallet-2.0.8/bin/mallet.bat'
```

Figure 120 Load Mallet library for LDA

```
In [12]: def compute_coherence_values(dictionary, corpus, texts, limit, start=2, step=3):
             """
             Compute c_v coherence for various number of topics

             Parameters:
             ----------
             dictionary : Gensim dictionary
             corpus : Gensim corpus
             texts : List of input texts
             limit : Max num of topics

             Returns:
             -------
             model_list : List of LDA topic models
             coherence_values : Coherence values corresponding to the LDA model with respective number of topics
             """
             coherence_values = []
             model_list = []
             for num_topics in range(start, limit, step):
                 model = gensim.models.wrappers.LdaMallet(mallet_path, corpus=corpus, num_topics=num_topics, id2word=id2word)
                 model_list.append(model)
                 coherencemodel = CoherenceModel(model=model, texts=texts, dictionary=dictionary, coherence='c_v')
                 coherence_values.append(coherencemodel.get_coherence())

             return model_list, coherence_values
```

Figure 121 Define Function to calculate Coherence Score

In [14]:
```
# Show graph
limit=40; start=2; step=6;
x = range(start, limit, step)
plt.plot(x, coherence_values)
plt.xlabel("Num Topics")
plt.ylabel("Coherence score")
plt.legend(("coherence_values"), loc='best')
plt.show()
```

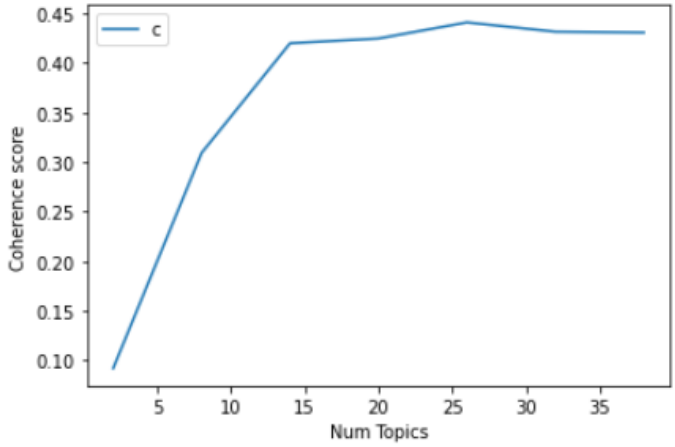


Figure 122 Display Coherence Score with Topic number

In [15]:
```
# Print the coherence scores
for m, cv in zip(x, coherence_values):
    print("Num Topics =", m, " has Coherence Value of", round(cv, 4))
```

```
Num Topics = 2  has Coherence Value of 0.0921
Num Topics = 8  has Coherence Value of 0.3092
Num Topics = 14  has Coherence Value of 0.4197
Num Topics = 20  has Coherence Value of 0.4245
Num Topics = 26  has Coherence Value of 0.4407
Num Topics = 32  has Coherence Value of 0.4313
Num Topics = 38  has Coherence Value of 0.4305
```

Figure 123 Output Coherence value with topic number

```
In [17]: # Select the model and print the topics ( num Topics = 14)
optimal_model = model_list[2]
model_topics = optimal_model.show_topics(formatted=False)
pprint(optimal_model.print_topics(num_words=10))
```

```
[(0,
  '0.038*"good" + 0.030*"week" + 0.024*"thing" + 0.021*"news" + 0.019*"back" + '
  '0.018*"put" + 0.015*"bad" + 0.015*"month" + 0.013*"day" + 0.013*"happen"'),
 (1,
  '0.024*"share" + 0.022*"update" + 0.021*"information" + 0.020*"follow" + '
  '0.020*"COVID" + 0.016*"late" + 0.015*"video" + 0.015*"free" + '
  '0.014*"resource" + 0.013*"learn"'),
 (2,
  '0.038*"Coronavirus" + 0.027*"quarantine" + 0.024*"find" + 0.022*"Corona" + '
  '0.019*"Covid" + 0.017*"Read" + 0.014*"person" + 0.014*"Virus" + '
  '0.013*"News" + 0.012*"xa0"'),
 (3,
  '0.086*"case" + 0.044*"day" + 0.033*"death" + 0.029*"report" + '
  '0.028*"number" + 0.027*"coronavirus" + 0.022*"year" + 0.022*"confirm" + '
  '0.018*"Italy" + 0.015*"today"'),
 (4,
  '0.032*"hand" + 0.018*"mask" + 0.018*"face" + 0.016*"hour" + 0.013*"wash" + '
  '0.012*"avoid" + 0.011*"run" + 0.010*"panic" + 0.010*"people" + '
  '0.010*"contact"'),
 (5,
  '0.042*"people" + 0.041*"health" + 0.040*"pandemic" + 0.033*"crisis" + '
  '0.033*"call" + 0.030*"care" + 0.021*"public" + 0.020*"risk" + '
  '0.020*"protect" + 0.020*"worker"'),
 (6,
  '0.090*"sign" + 0.083*"petition" + 0.079*"give" + 0.057*"light" + '
  '0.054*"basic" + 0.054*"income" + 0.051*"universal" + 0.038*"rent" + '
  '0.035*"join" + 0.035*"suspend"'),
 (7,
  '0.047*"people" + 0.046*"virus" + 0.040*"world" + 0.027*"life" + '
  '0.021*"fight" + 0.013*"love" + 0.013*"hope" + 0.013*"make" + 0.012*"save" + '
  '0.011*"talk"'),
 (8,
  '0.068*"test" + 0.054*"COVID" + 0.034*"patient" + 0.029*"March" + '
  '0.022*"hospital" + 0.020*"Coronavirus" + 0.013*"PM" + 0.012*"Health" + '
  '0.011*"pm" + 0.010*"result"'),
 (9,
  '0.028*"close" + 0.027*"business" + 0.022*"plan" + 0.018*"service" + '
  '0.017*"due" + 0.016*"shut" + 0.016*"support" + 0.015*"impact" + '
  '0.014*"order" + 0.013*"school"'),
 (10,
  '0.060*"spread" + 0.041*"China" + 0.039*"country" + 0.031*"coronavirus" + '
  '0.027*"outbreak" + 0.022*"measure" + 0.015*"Chinese" + 0.015*"India" + '
  '0.012*"citizen" + 0.012*"lockdown"'),
 (11,
  '0.025*"Trump" + 0.023*"time" + 0.020*"show" + 0.018*"symptom" + '
  '0.018*"medical" + 0.014*"stop" + 0.013*"supply" + 0.012*"Dr" + '
  '0.012*"office" + 0.012*"treatment"'),
 (12,
  '0.023*"make" + 0.016*"disease" + 0.016*"change" + 0.015*"read" + '
  '0.015*"response" + 0.012*"situation" + 0.011*"long" + 0.010*"understand" + '
  '0.010*"article" + 0.010*"gt"'),
 (13,
  '0.065*"home" + 0.065*"work" + 0.057*"time" + 0.043*"stay" + 0.036*"safe" + '
  '0.027*"Stay" + 0.026*"distance" + 0.026*"social" + 0.020*"family" + '
  '0.017*"continue"')]
```

Figure 124 Display topics for optimal model

```
In [19]: # convert the class of your mallet model into a LdaModel before pyLDAvis
         model = gensim.models.wrappers.ldamallet.malletmodel2ldamodel(optimal_model)
```

Figure 125 Convert mallet model to LDA model

```
In [21]: # Visualise the topic keyword
         pyLDAvis.enable_notebook()
         vis = pyLDAvis.gensim_models.prepare(model, corpus, id2word)
         vis
```

Figure 126 Display the optimal model

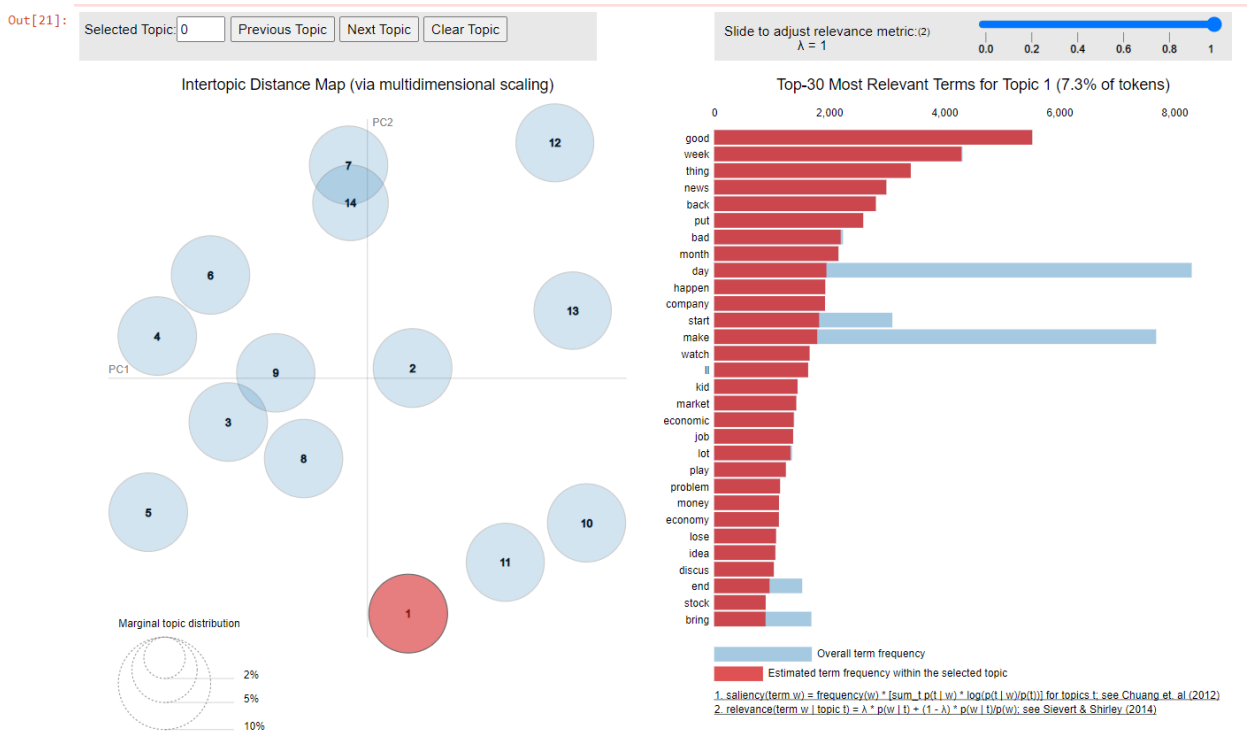


Figure 127 Display the Intertopic Distance Map

I used the below approach to find he dominate topic for each sentence as Figure 128 and Figure 129

```
In [22]: # Finding the dominant topic in each sentence
def format_topics_sentences(ldamodel=lda_model, corpus=corpus, texts=lem):
    # Init output
    sent_topics_df = pd.DataFrame()

    # Get main topic in each document
    for i, row in enumerate(ldamodel[corpus]):
        row = sorted(row[0], key=lambda x: (x[1]), reverse=True)
        # Get the Dominant topic, Perc Contribution and Keywords for each document
        for j, (topic_num, prop_topic) in enumerate(row):
            if j == 0:  # => dominant topic
                wp = ldamodel.show_topic(topic_num)
                topic_keywords = ", ".join([word for word, prop in wp])
                sent_topics_df = sent_topics_df.append(pd.Series([int(topic_num), round(prop_topic,4), topic_keywords]), ignore_index=True)
            else:
                break
    sent_topics_df.columns = ['Dominant_Topic', 'Perc_Contribution', 'Topic_Keywords']

    # Add original text to the end of the output
    contents = pd.Series(texts)
    sent_topics_df = pd.concat([sent_topics_df, contents], axis=1)
    return(sent_topics_df)


df_topic_sents_keywords = format_topics_sentences(ldamodel=lda_model, corpus=corpus, texts=lem)

# Format
df_dominant_topic = df_topic_sents_keywords.reset_index()
df_dominant_topic.columns = ['Document_No', 'Dominant_Topic', 'Topic_Perc_Contrib', 'Keywords', 'Text']

# Show
df_dominant_topic.head(10)
```

Figure 128 Find the dominant topic in each sentence

Out[22]:

	Document_No	Dominant_Topic	Topic_Perc_Contrib	Keywords	Text
0	0	0.0	0.3666	case, COVID, Coronavirus, new, update, come, t...	['Italy', 'report', 'first', 'locally', 'trans...
1	1	3.0	0.3440	everyone, people, get, test, day, The, u, viru...	['quarantine', 'Hubu', 'Village', 'Shuangpu', ...
2	2	3.0	0.3170	everyone, people, get, test, day, The, u, viru...	['Italy', 'love', 'country']
3	3	3.0	0.4130	everyone, people, get, test, day, The, u, viru...	['dispersion', 'globallyn', 'Analysis', 'trave...
4	4	1.0	0.3680	ask, sign, petition, income, basic, universal,...	['This']
5	5	1.0	0.4922	ask, sign, petition, income, basic, universal,...	['warn', 'community', 'spread', 'could', 'take...
6	6	3.0	0.3969	everyone, people, get, test, day, The, u, viru...	['After', 'thorough', 'investigation', 'time',...
7	7	2.0	0.4020	give, light, like, know, good, one, close, thi...	['look', 'like', 'force', 'removal', 'isolatio...
8	8	3.0	0.4125	everyone, people, get, test, day, The, u, viru...	['CDC', 'didn', 'want', 'patient', 'fly', 'US'...
9	9	3.0	0.3039	everyone, people, get, test, day, The, u, viru...	['N100', 'Disposable', 'Respirator', 'Cool', '...

Figure 129 Display the dominant topic in each sentence

I ran the below procedure to find he most representative document for each topic as Figure 130 and Figure 131

```
In [23]: #Find the most representative document for each topic
         # Group top 5 sentences under each topic
         sent_topics_sorteddf_mallet = pd.DataFrame()

         sent_topics_outdf_grpd = df_topic_sents_keywords.groupby('Dominant_Topic')

         for i, grp in sent_topics_outdf_grpd:
             sent_topics_sorteddf_mallet = pd.concat([sent_topics_sorteddf_mallet,
                                                      grp.sort_values(['Perc_Contribution'], ascending=[0]).head(1)],
                                                     axis=0)

         # Reset Index
         sent_topics_sorteddf_mallet.reset_index(drop=True, inplace=True)

         # Format
         sent_topics_sorteddf_mallet.columns = ['Topic_Num', "Topic_Perc_Contrib", "Keywords", "Text"]

         # Show
         sent_topics_sorteddf_mallet.head()
```

Figure 130 Find the most representative document for each topic

Out[23]:

	Topic_Num	Topic_Perc_Contrib	Keywords	Text
0	0.0	0.6557	case, COVID, Coronavirus, new, update, come, t...	['Report', 'China', 'New', 'Cases', 'New', 'De...
1	1.0	0.6664	ask, sign, petition, income, basic, universal,...	['We', 'need', 'plan', 'renter', 'impact', 'Si...
2	2.0	0.6685	give, light, like, know, good, one, close, thi...	['Chinese', 'Virus', 'Chinese', 'Virus', 'Chin...
3	3.0	0.7904	everyone, people, get, test, day, The, u, viru...	['STAY', 'HOME', 'STAY', 'HOME', 'STAY', 'HOME...

Figure 131 Display the most representative document for each topic

The below procedure was run to display the most representative document for the topic

```
In [24]: #Topic distribution across documents

         # Number of Documents for Each Topic
         topic_counts = df_topic_sents_keywords['Dominant_Topic'].value_counts()

         # Percentage of Documents for Each Topic
         topic_contribution = round(topic_counts/topic_counts.sum(), 4)

         # Topic Number and Keywords
         topic_num_keywords = df_topic_sents_keywords[['Dominant_Topic', 'Topic_Keywords']]

         # Concatenate Column wise
         df_dominant_topics = pd.concat([topic_num_keywords, topic_counts, topic_contribution], axis=1)

         # Change Column names
         df_dominant_topics.columns = ['Dominant_Topic', 'Topic_Keywords', 'Num_Documents', 'Perc_Documents']

         # Show
         df_dominant_topics
```

Figure 132 Find Topic distribution across documents

Out[24]:

	Dominant_Topic	Topic_Keywords	Num_Documents	Perc_Documents
0.0	0.0	case, COVID, Coronavirus, new, update, come, t...	7563.0	0.0757
1.0	3.0	everyone, people, get, test, day, The, u, viru...	58221.0	0.5829
2.0	3.0	everyone, people, get, test, day, The, u, viru...	16596.0	0.1662
3.0	3.0	everyone, people, get, test, day, The, u, viru...	17505.0	0.1753
4.0	1.0	ask, sign, petition, income, basic, universal,...	NaN	NaN
...	...	...	...	...
99880.0	1.0	ask, sign, petition, income, basic, universal,...	NaN	NaN
99881.0	1.0	ask, sign, petition, income, basic, universal,...	NaN	NaN
99882.0	2.0	give, light, like, know, good, one, close, thi...	NaN	NaN
99883.0	1.0	ask, sign, petition, income, basic, universal,...	NaN	NaN
99884.0	3.0	everyone, people, get, test, day, The, u, viru...	NaN	NaN

99885 rows × 4 columns

Figure 133 Display Topic distribution across documents

5.1.2 Reddit Data

The process for Reddit Data is similar to Twitter as Figure 134 to Figure 143

In [1]:
```
import gensim
import gensim.corpora as corpora
from gensim.corpora import Dictionary
from gensim.models.coherencemodel import CoherenceModel
from gensim.models.ldamodel import LdaModel

from pprint import pprint

import spacy

import pickle
import re
import pyLDAvis
import pyLDAvis.gensim_models

import matplotlib.pyplot as plt
import pandas as pd
```

```
C:\Users\weihe\Anaconda3\lib\site-packages\gensim\similarities\__init__.py:15: UserWarning: The g
evenshtein package <https://pypi.org/project/python-Levenshtein/> is unavailable. Install Levenhs
  warnings.warn(msg)
```

In [2]:
```
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)
```

Figure 134 Load library

```
In [3]: df = pd.read_csv("export_reddit_dataset.csv")
        df = df.drop('origin', 1)
        df.head()
```

```
C:\Users\weihe\AppData\Local\Temp/ipykernel_29772/1506153113.py:2: FutureWarning: In a future version of pandas all arguments o
f DataFrame.drop except for the argument 'labels' will be keyword-only
  df = df.drop('origin', 1)
```

Out[3]:

	id	Score	timestamp	comments	text	sentiment	Tokenized_Reddit	lemmatized_text
0	f2kpf6	1	2020-02-12 03:28:58	0	found something lancet regarding cytokine stor...	0.000000	['found', 'something', 'lancet', 'regarding', ...	['find', 'something', 'lancet', 'regard', 'cyt...
1	f2m2cv	1	2020-02-12 05:13:07	5	info covid affect pet domestic animal	0.000000	['info', 'covid', 'affect', 'pet', 'domestic',...	['info', 'covid', 'affect', 'pet', 'domestic',...
2	f2p51b	1	2020-02-12 10:16:11	0	john hopkins bloomberg school public health co...	0.000000	['john', 'hopkins', 'bloomberg', 'school', 'pu...	['john', 'hopkins', 'bloomberg', 'school', 'pu...
3	f2qh4a	1	2020-02-12 12:28:14	0	charge victory smash covid virus wipe bad viru...	-0.700000	['charge', 'victory', 'smash', 'covid', 'virus...	['charge', 'victory', 'smash', 'covid', 'virus...
4	f2roo5	1	2020-02-12 14:04:57	5	china report smallest number new covid case s...	-0.088068	['china', 'report', 'smallest', 'number', 'new...	['china', 'report', 'small', 'number', 'new', ...

Figure 135 Load data

```
In [4]: lem = df['lemmatized_text'].values.tolist()

        lemmatized_list = []
        for lemmatized_item in lem:
            print(lemmatized_item)
            lemmatized_list.append(eval(lemmatized_item))

        print(lemmatized_list[:2])
```

```
['two', 'data', 'point', 'singapore', 'covid', 'case', 'data']
['excellent', 'database', 'covid']
['new', 'england', 'journal', 'medicine', 'define', 'epidemiology', 'covid', 'study', 'need']
['excellent', 'database', 'covid', 'covid19', 'covid', 'novel', 'coronavirus', 'ncov', 'ncov', 'sars', 'cov', 'sars2', 'wuha
n', 'virus', 'statistic', 'statistical', 'graph', 'data', 'analysis', 'live', 'spreadsheet']
['myth', 'buster', 'covid']
['field', 'brief', 'diamond', 'princess', 'covid', 'case']
['longitudinal', 'ct', 'find', 'covid', 'pneumonia', 'case', 'present', 'organize', 'pneumonia', 'pattern', 'case', 'includ
e', 'relapse', 'successfully', 'resolve']
['update', 'covid', 'outbreak', 'professor', 'neil', 'ferguson', 'dr', 'erik', 'volz']
['novel', 'coronavirus', 'covid', 'outbreak', 'review', 'current', 'literature']
['live', 'vr', 'lecture', 'covid', 'molecular', 'structure']
['complete', 'covid', 'multisub', 'yet', 'please', 'let', 'know', 'miss', 'anything', 'want', 'multi', 'inclusive', 'possibl
e']
['covid', 'originate', 'wuhan', 'seafood', 'market', 'say', 'american', 'senator']
['coronavirus', 'disease', 'covid', 'situation', 'report']
['police', 'quarantine', 'whole', 'community', 'beijing', 'confirm', 'covid', 'case']
['covid', 'singapore', 'current', 'experience']
['japan', 'national', 'institute', 'infectious', 'disease', 'field', 'brief', 'diamond', 'princess', 'covid', 'case']
['coronavirus', 'disease', 'covid', 'situation', 'report']
```

Figure 136 Extract the lemmatized_text feature

```
In [5]: # Create the Dictionary (id2word) and Corpus needed for Topic Modeling

        # Create the dictionary
        id2word = corpora.Dictionary(lemmatized_list)

        # Create corpus
        texts = lemmatized_list

        # Term Document Frequency
        corpus = [id2word.doc2bow(text) for text in texts]

        #View
        print(corpus[:1])
```

```
[[(0, 1), (1, 1), (2, 1), (3, 1), (4, 1), (5, 1), (6, 1), (7, 1), (8, 1), (9, 1), (10, 1)]]
```

Figure 137 Compute the id2word dictionary

```
In [6]: # Pass the id as a key to the dictionary to see which word is associated to the given id
        id2word[0]

Out[6]: 'article'
```

Figure 138 Display the id2word dictionary

```
In [7]: #Build Topic model

        #Here we supply the corpus, dictionary, number of topics
        #chunksize presents the number of documents used for each traning chunk
        #update_every presents the frequency that the model should be updated
        #passed presents the total number of training passes


        lda_model = gensim.models.ldamodel.LdaModel(corpus=corpus,
                                                    id2word=id2word,
                                                    num_topics=10,
                                                    random_state=100,
                                                    update_every=1,
                                                    chunksize=100,
                                                    passes=10,
                                                    alpha='auto',
                                                    per_word_topics=True)
```

Figure 139 Build the base model with 10 topics

```
In [8]: pprint(lda_model.print_topics())
        doc_lda=lda_model[corpus]

        [(0,
          '0.072*"covid" + 0.052*"case" + 0.042*"death" + 0.033*"country" + '
          '0.029*"coronavirus" + 0.024*"say" + 0.024*"spread" + 0.019*"germany" + '
          '0.018*"report" + 0.018*"show"'),
         (1,
          '0.051*"stay" + 0.043*"trump" + 0.037*"link" + 0.037*"amp" + 0.029*"comment" '
          '+ 0.029*"watch" + 0.029*"around" + 0.028*"guy" + 0.028*"kill" + '
          '0.025*"tip"'),
         (2,
          '0.078*"pandemic" + 0.057*"make" + 0.024*"know" + 0.022*"stop" + '
          '0.022*"please" + 0.021*"cure" + 0.021*"create" + 0.017*"research" + '
          '0.017*"top" + 0.017*"post"'),
         (3,
          '0.062*"world" + 0.045*"live" + 0.040*"day" + 0.033*"news" + 0.025*"may" + '
          '0.019*"uk" + 0.019*"indian" + 0.018*"call" + 0.017*"end" + 0.015*"warn"'),
         (4,
          '0.043*"lockdown" + 0.035*"die" + 0.032*"way" + 0.032*"song" + 0.031*"life" '
          '+ 0.030*"number" + 0.025*"safe" + 0.022*"good" + 0.022*"best" + '
          '0.021*"year"'),
         (5,
          '0.066*"crisis" + 0.059*"patient" + 0.024*"see" + 0.019*"situation" + '
          '0.019*"cause" + 0.018*"even" + 0.017*"much" + 0.015*"write" + 0.014*"amid" '
          '+ 0.014*"blood"'),
         (6,
          '0.473*"corona" + 0.206*"virus" + 0.011*"sweden" + 0.010*"first" + '
          '0.009*"like" + 0.008*"hospital" + 0.008*"take" + 0.006*"name" + '
          '0.005*"quarantine" + 0.005*"prevent"'),
         (7,
          '0.048*"new" + 0.041*"update" + 0.038*"outbreak" + 0.032*"video" + '
          '0.032*"fight" + 0.030*"china" + 0.025*"find" + 0.023*"mask" + 0.020*"use" + '
          '0.019*"health"'),
         (8,
          '0.052*"people" + 0.048*"get" + 0.044*"time" + 0.041*"help" + 0.036*"german" '
          '+ 0.030*"go" + 0.029*"due" + 0.025*"infect" + 0.021*"could" + 0.021*"one"'),
         (9,
          '0.067*"india" + 0.064*"test" + 0.041*"vaccine" + 0.027*"think" + '
          '0.025*"home" + 0.021*"positive" + 0.020*"give" + 0.015*"put" + '
          '0.013*"start" + 0.013*"come"')]
```

Figure 140 Display the generated topics

```
In [9]: print('Perplexity: ', lda_model.log_perplexity(corpus))

        # Compute Coherence Score
        coherence_model_lda = CoherenceModel(model=lda_model, texts=lemmatized_list, dictionary=id2word, coherence='c_v')
        coherence_lda = coherence_model_lda.get_coherence()
        print('Coherence Score: ', coherence_lda)

        Perplexity:  -8.794595465509639
        Coherence Score:  0.37826766431123093
```

Figure 141 Compute the Perplexity and Coherence score

```
In [10]: # Visualise the topic keyword
         pyLDAvis.enable_notebook()
         vis = pyLDAvis.gensim_models.prepare(lda_model, corpus, id2word)
         vis

         C:\Users\weihe\Anaconda3\lib\site-packages\pyLDAvis\_prepare.py:246: FutureWarning
         bels' will be keyword-only
           default_term_info = default_term_info.sort_values(
```

Figure 142 View the model

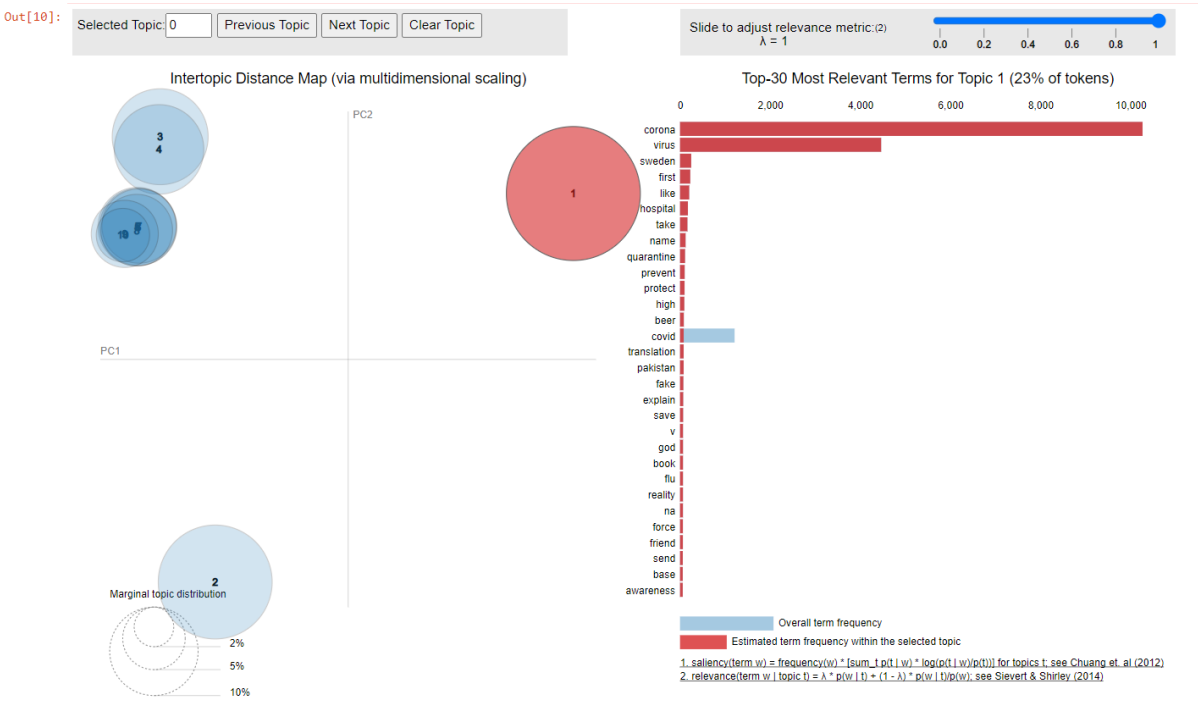


Figure 143 Display of the model

5.2 Sentiment Analysis

5.2.1 Twitter Data

I used the TextBlob to calculate the sentiment after preprocessing as Figure 144 to Figure 148

In [1]:
```
#Sentiment Analysis
import os
import pandas as pd
from textblob import TextBlob
import numpy as np

# For consusion matrix and metrics
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report

#For statistics
import statistics
import math
from math import sqrt
from numpy import mean
from scipy.stats import t

# For Visualisation
from matplotlib import pyplot as plt
import seaborn as sns

# For T-test using spicy
from numpy.random import seed
from numpy.random import randn
from scipy.stats import ttest_rel
from scipy.stats import ttest_ind
```

Figure 144 Load library

In [2]:
```
df = pd.read_csv('tweet_dataframe_process.csv')
df.head()
```

Out[2]:

	id	lang	timestamp	origin	author	rt	fav	text	sentiment	Tokenized_Tweet	lemmatized_text
0	1231005111071100929	en	2020-02-21T23:57:44.000Z	b'Italy \\U0001f1ee\\U0001f1f9 reports first l...	2985110557	2	1	Italy reports first locally transmitted c...	0.083333	['Italy', 'reports', 'first', 'locally', 'tran...	['Italy', 'report', 'first', 'locally', 'trans...
1	1231005062165458945	en	2020-02-21T23:57:32.000Z	b'#Coronavirus quarantine in a Hubu Village, S...	1202960464549838848	1	2	quarantine Hubu Village Shuangpu Town Xihu D...	0.000000	['quarantine', 'Hubu', 'Village', 'Shuangpu', ...	['quarantine', 'Hubu', 'Village', 'Shuangpu', ...
2	1231004398811275264	en	2020-02-21T23:54:54.000Z	b'Italy. I love this country. #coronavirusital...	1221955502340558848	0	3	Italy love country	0.500000	['Italy', 'love', 'country']	['Italy', 'love', 'country']
3	1230866203947798530	en	2020-02-21T23:53:24.000Z	b'@JoannaB36464141 @nicolamlow @alexvespi @Ada...	1093695069176000512	0	2	dispersion globallyn Analysis travel Wuhan av...	0.400000	['dispersion', 'globallyn', 'Analysis', 'trave...	['dispersion', 'globallyn', 'Analysis', 'trave...
4	1231003954806501376	en	2020-02-21T23:53:08.000Z	b'@WHO \\nThis is a #pandemic \\n#CoronavirusO...	723319704710844417	0	0	This	0.000000	['This']	['This']

Figure 145 Load Data

```
In [4]: sentiment2_class = []

for index, score in df.iterrows():
    score = score['sentiment2']
    if score > 0.3:
        score_class = 1 # positive
    elif score < -0.3:
        score_class = -1 # negative
    else:
        score_class = 0
    sentiment2_class.append(score_class)

df['sentiment2_class'] = sentiment2_class #create new colume in df with output

sentiment1_class = []

for index, score in df.iterrows():
    score = score['sentiment']
    if score > 0.3:
        score_class = 1  # positive
    elif score <-0.3:
        score_class = -1  # negative
    else:
        score_class = 0  # neutral
    sentiment1_class.append(score_class)

df['sentiment1_class'] = sentiment1_class # print values and add the colume in df
```

Figure 146 Create Sentiment classification

```
In [5]: print('Confusion Matrix : ')
        print(confusion_matrix(sentiment1_class, sentiment2_class))
        print('Accuracy Score : ', accuracy_score(sentiment1_class, sentiment2_class))
        print('Classification Report : ')
        print(classification_report(sentiment1_class, sentiment2_class))

Confusion Matrix :
[[  55534    8794     106]
 [  13785 1263696   23667]
 [    321   35107  227404]]
Accuracy Score :  0.9497793558640493
Classification Report :
              precision    recall  f1-score   support

          -1       0.80      0.86      0.83     64434
           0       0.97      0.97      0.97   1301148
           1       0.91      0.87      0.88    262832

    accuracy                           0.95   1628414
   macro avg       0.89      0.90      0.89   1628414
weighted avg       0.95      0.95      0.95   1628414
```

Figure 147 Build Confusion matrix

```
In [6]: #Count of the occurances of each of the unique values in the columns stated
        print('Count in Sentiment2 class (computed after pre-processing text)')
        print(df['sentiment2_class'].value_counts())
        print('Counts in Sentiment1 class (computed on raw text)')
        print(df['sentiment1_class'].value_counts())
```

```
Count in Sentiment2 class (computed after pre-processing text)
 0    1307597
 1     251177
-1      69640
Name: sentiment2_class, dtype: int64
Counts in Sentiment1 class (computed on raw text)
 0    1301148
 1     262832
-1      64434
Name: sentiment1_class, dtype: int64
```

Figure 148 Count for the occurance of each category

I compared the difference before and after pre-processing as Figure 149 to Figure 152

```
In [8]: x1 = df.loc[df.sentiment1_class==1, 'sentiment']
        x2 = df.loc[df.sentiment1_class==0, 'sentiment']
        x3 = df.loc[df.sentiment1_class==-1, 'sentiment']

        kwargs = dict(alpha=0.5, bins=5, density=True, stacked=True)

        plt.hist(x1, **kwargs, color='g', label='Positive')
        plt.hist(x2, **kwargs, color='y', label='Neutral')
        plt.hist(x3, **kwargs, color='r', label='Negative')

        #Normalized Frequency sentiment1
        plt.gca().set(title='Tweet Frequency Histogram of Sentiment1 Score', ylabel='Tweet Frequecy')
        plt.xlim(-1,1)
        plt.legend()

Out[8]: <matplotlib.legend.Legend at 0x1e613324070>
```

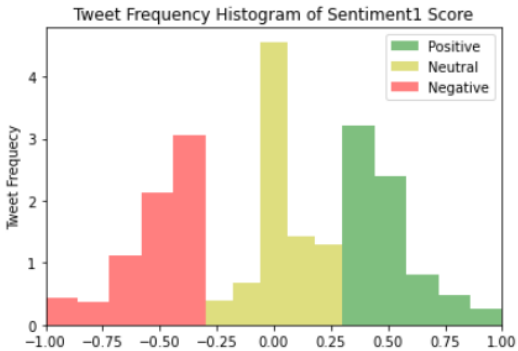


Figure 149 Display the distribution before pre-processing

```
In [9]: x1 = df.loc[df.sentiment2_class==1, 'sentiment2']
        x2 = df.loc[df.sentiment2_class==0, 'sentiment2']
        x3 = df.loc[df.sentiment2_class==-1, 'sentiment2']

        kwargs = dict(alpha=0.5, bins=5, density=True, stacked=True)

        plt.hist(x1, **kwargs, color='g', label='Positive')
        plt.hist(x2, **kwargs, color='y', label='Neutral')
        plt.hist(x3, **kwargs, color='r', label='Negative')

        #Normalized Frequency sentiment2
        plt.gca().set(title='Tweet Frequency Histogram of Sentiment2 Score', ylabel='Tweet Frequecy')
        plt.xlim(-1,1)
        plt.legend()

Out[9]: <matplotlib.legend.Legend at 0x1e61332a100>
```

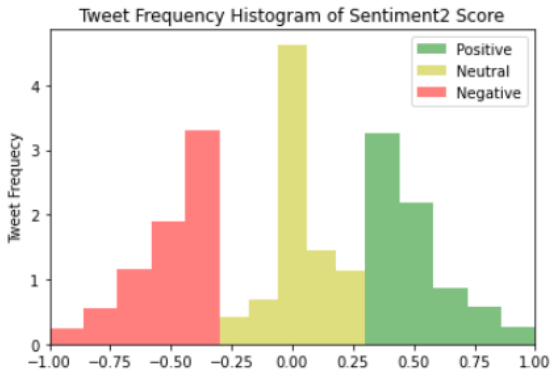


Figure 150 Display the distribution after pre-processing

```
In [11]: # sentiment1 score plotted per class
         sns.catplot(x="sentiment1_class", y="sentiment",data=df, jitter='0.4')
         plt.title('Sentiment1 scores per Class')

Out[11]: Text(0.5, 1.0, 'Sentiment1 scores per Class')
```

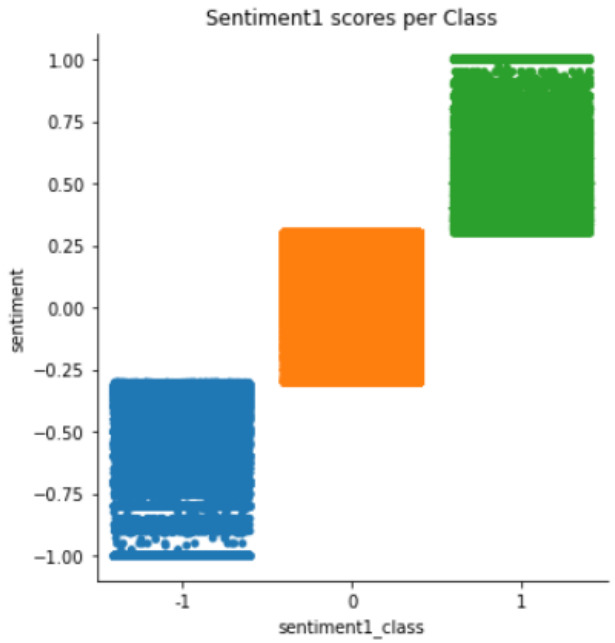


Figure 151 Display the sentiment before pre-processing

```
In [12]: # sentiment2 score plotted per class
         sns.catplot(x="sentiment2_class", y="sentiment2",data=df, jitter='0.4')
         plt.title('Sentiment2 scores per Class')

Out[12]: Text(0.5, 1.0, 'Sentiment2 scores per Class')
```

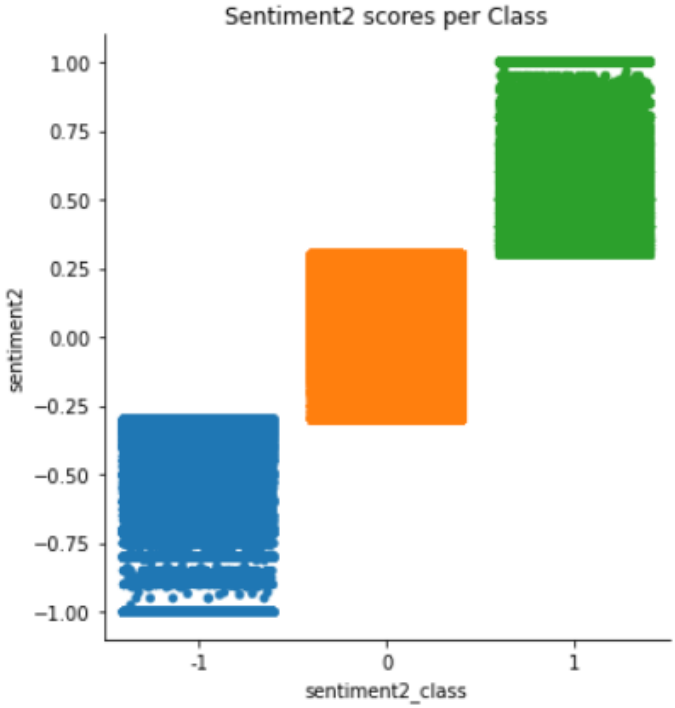


Figure 152 Display the sentiment after pre-processing

I generated the paired T- test to compare the difference in sentiment as Figure 153 and Figure 154

```
In [13]: # Paied t-test with spicy library

         # Generate 2 independent samples
         data1 = df['sentiment'].values.tolist()
         data2 = df['sentiment2'].values.tolist()

         stat, p = ttest_rel(data1, data2)
         print('Statistics=%.3f, p=%.3f' % (stat,p))

         Statistics=98.349, p=0.000
```

Figure 153 Generate Paired t-test result

```
In [14]: # Welch's non parameteric t-test
         data_tweets = df['sentiment2'].values.tolist()

         print('Variance sentiment2 : ', round(statistics.stdev(data_tweets), 3))

         Variance sentiment2 :  0.245
```

Figure 154 Generate non parametric t-test result

```
In [15]: df.to_csv(r'C:\Users\weihe\Documents\TweetData\twitter_sentiment.csv', index = False, header = True)
```

Figure 155 Store the dataset with new sentiment field added

5.2.2 Reddit Data

I have processed the Reddit Data with the same approach as Figure 156 to Figure 169.

```
In [1]: #Sentiment Analysis
        import os
        import pandas as pd
        from textblob import TextBlob
        import numpy as np

        # For consusion matrix and metrics
        from sklearn.metrics import confusion_matrix
        from sklearn.metrics import accuracy_score
        from sklearn.metrics import classification_report

        #For statistics
        import statistics
        import math
        from math import sqrt
        from numpy import mean
        from scipy.stats import t

        # For Visualisation
        from matplotlib import pyplot as plt
        import seaborn as sns

        # For T-test using spicy
        from numpy.random import seed
        from numpy.random import randn
        from scipy.stats import ttest_rel
        from scipy.stats import ttest_ind
```

Figure 156 Load library

```
In [2]: df = pd.read_csv('export_reddit_dataset.csv')
        df.head()
```

Out[2]:

	id	origin	Score	timestamp	comments	text	sentiment	Tokenized_Reddit	lemmatized_text
0	f2kpf6	Found something in "The Lancet" regarding cyto...	1	2020-02-12 03:28:58	0	found something lancet regarding cytokine stor...	0.000000	['found', 'something', 'lancet', 'regarding', ...	['find', 'something', 'lancet', 'regard', 'cyt...
1	f2m2cv	Any info on how COVID-19 affect pets/domestic ...	1	2020-02-12 05:13:07	5	info covid affect pet domestic animal	0.000000	['info', 'covid', 'affect', 'pet', 'domestic',...	['info', 'covid', 'affect', 'pet', 'domestic',...
2	f2p51b	Johns Hopkins Bloomberg school of public healt...	1	2020-02-12 10:16:11	0	john hopkins bloomberg school public health co...	0.000000	['john', 'hopkins', 'bloomberg', 'school', 'pu...	['john', 'hopkins', 'bloomberg', 'school', 'pu...
3	f2qh4a	Charge for Victory!!! Smash down the COVID vir...	1	2020-02-12 12:28:14	0	charge victory smash covid virus wipe bad viru...	-0.700000	['charge', 'victory', 'smash', 'covid', 'virus...	['charge', 'victory', 'smash', 'covid', 'virus...
4	f2roo5	China Reports Smallest Number Of New COVID-19 ...	1	2020-02-12 14:04:57	5	china report smallest number new covid case s...	-0.088068	['china', 'report', 'smallest', 'number', 'new...	['china', 'report', 'small', 'number', 'new', ...

Figure 157 Load data

```
In [3]: df[['sentiment2', 'subjective']] = df['lemmatized_text'].apply(lambda lemmatized_text: pd.Series(TextBlob(lemmatized_text).sentim
```

Figure 158 Create Sentiment2 field

```
In [4]: sentiment2_class = []

for index, score in df.iterrows():
    score = score['sentiment2']
    if score > 0.3:
        score_class = 1 # positive
    elif score < -0.3:
        score_class = -1 # negative
    else:
        score_class = 0
    sentiment2_class.append(score_class)

df['sentiment2_class'] = sentiment2_class #create new colume in df with output

sentiment1_class = []

for index, score in df.iterrows():
    score = score['sentiment']
    if score > 0.3:
        score_class = 1  # positive
    elif score <-0.3:
        score_class = -1  # negative
    else:
        score_class = 0  # neutral
    sentiment1_class.append(score_class)

df['sentiment1_class'] = sentiment1_class # print values and add the colume in df
```

Figure 159 Create Sentiment class

```
In [5]: print('Confusion Matrix : ')
        print(confusion_matrix(sentiment1_class, sentiment2_class))
        print('Accuracy Score : ', accuracy_score(sentiment1_class, sentiment2_class))
        print('Classification Report : ')
        print(classification_report(sentiment1_class, sentiment2_class))

Confusion Matrix :
[[ 265   42    0]
 [  90 9173   59]
 [   3  205  804]]
Accuracy Score :  0.9625035241048774
Classification Report :
              precision    recall  f1-score   support

          -1       0.74      0.86      0.80       307
           0       0.97      0.98      0.98      9322
           1       0.93      0.79      0.86      1012

    accuracy                           0.96     10641
   macro avg       0.88      0.88      0.88     10641
weighted avg       0.96      0.96      0.96     10641
```

Figure 160 Print Confusion Matrix

```
In [6]: #Count of the occurances of each of the unique values in the columns stated
        print('Count in Sentiment2 class (computed after pre-processing text)')
        print(df['sentiment2_class'].value_counts())
        print('Counts in Sentiment1 class (computed on raw text)')
        print(df['sentiment1_class'].value_counts())

Count in Sentiment2 class (computed after pre-processing text)
 0    9420
 1     863
-1     358
Name: sentiment2_class, dtype: int64
Counts in Sentiment1 class (computed on raw text)
 0    9322
 1    1012
-1     307
Name: sentiment1_class, dtype: int64
```

Figure 161 Output occurance of each value

```
In [7]: df.head()
```

Out[7]:

	id	origin	Score	timestamp	comments	text	sentiment	Tokenized_Reddit	lemmatized_text	sentiment2	subjective	sentiment2_class	sent
0	f2kpf6	Found something in "The Lancet" regarding cyto...	1	2020-02-12 03:28:58	0	found something lancet regarding cytokine stor...	0.000000	['found', 'something', 'lancet', 'regarding', ...	['find', 'something', 'lancet', 'regard', 'cyt...	0.000000	0.000000	0	
1	f2m2cv	Any info on how COVID-19 affect pets/domestic ...	1	2020-02-12 05:13:07	5	info covid affect pet domestic animal	0.000000	['info', 'covid', 'affect', 'pet', 'domestic',...	['info', 'covid', 'affect', 'pet', 'domestic',...	0.000000	0.100000	0	
2	f2p51b	Johns Hopkins Bloomberg school of	1	2020-02-12 10:16:11	0	john hopkins bloomberg school public	0.000000	['john', 'hopkins', 'bloomberg', 'school', 'pu...	['john', 'hopkins', 'bloomberg', 'school', 'pu...	0.000000	0.066667	0	

Figure 162 Display the data

```
In [8]: x1 = df.loc[df.sentiment1_class==1, 'sentiment']
        x2 = df.loc[df.sentiment1_class==0, 'sentiment']
        x3 = df.loc[df.sentiment1_class==-1, 'sentiment']

        kwargs = dict(alpha=0.5, bins=5, density=True, stacked=True)

        plt.hist(x1, **kwargs, color='g', label='Positive')
        plt.hist(x2, **kwargs, color='y', label='Neutral')
        plt.hist(x3, **kwargs, color='r', label='Negative')

        #Normalized Frequency sentiment1
        plt.gca().set(title='Tweet Frequency Histogram of Sentiment1 Score', ylabel='Tweet Fr
        plt.xlim(-1,1)
        plt.legend()

Out[8]: <matplotlib.legend.Legend at 0x25b5dd3c850>
```

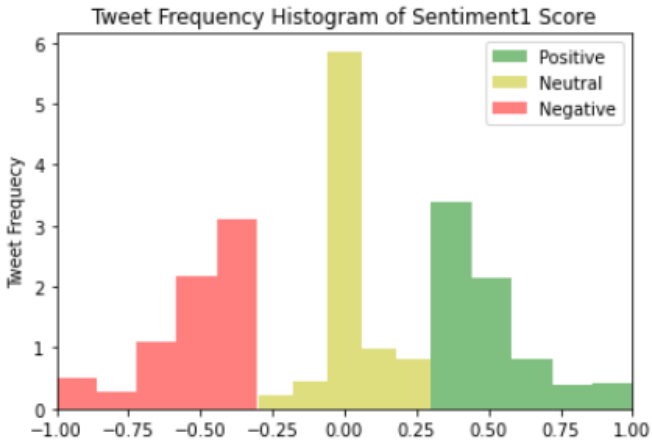


Figure 163 Display plot for sentiment before pre-processing

```
In [9]: x1 = df.loc[df.sentiment2_class==1, 'sentiment2']
        x2 = df.loc[df.sentiment2_class==0, 'sentiment2']
        x3 = df.loc[df.sentiment2_class==-1, 'sentiment2']

        kwargs = dict(alpha=0.5, bins=5, density=True, stacked=True)

        plt.hist(x1, **kwargs, color='g', label='Positive')
        plt.hist(x2, **kwargs, color='y', label='Neutral')
        plt.hist(x3, **kwargs, color='r', label='Negative')

        #Normalized Frequency sentiment2
        plt.gca().set(title='Tweet Frequency Histogram of Sentiment2 Score', ylabel='Tweet Frequecy')
        plt.xlim(-1,1)
        plt.legend()

Out[9]: <matplotlib.legend.Legend at 0x25b5dd4fca0>
```

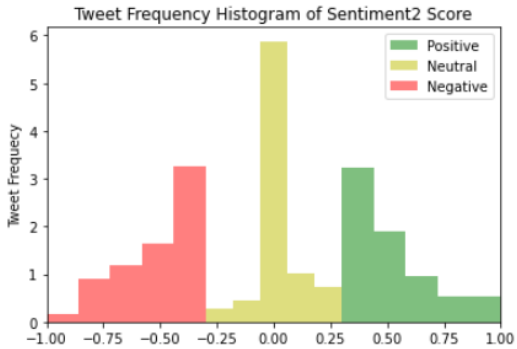


Figure 164 Display plot for sentiment after pre-processing

```
In [10]: # sentiment1 score plotted per class
         sns.catplot(x="sentiment1_class", y="sentiment",data=df, jitter='0.4')
         plt.title('Sentiment1 scores per Class')

Out[10]: Text(0.5, 1.0, 'Sentiment1 scores per Class')
```

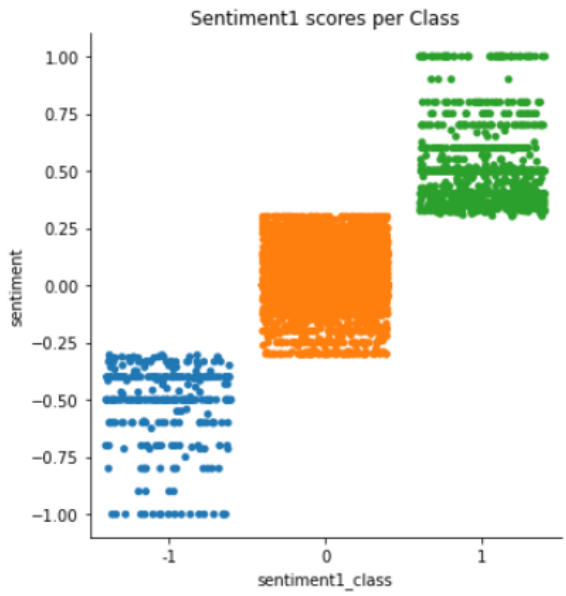


Figure 165 Display sentiment score before preprocessing

```
In [11]: # sentiment2 score plotted per class
         sns.catplot(x="sentiment2_class", y="sentiment2",data=df, jitter='0.4')
         plt.title('Sentiment2 scores per Class')

Out[11]: Text(0.5, 1.0, 'Sentiment2 scores per Class')
```

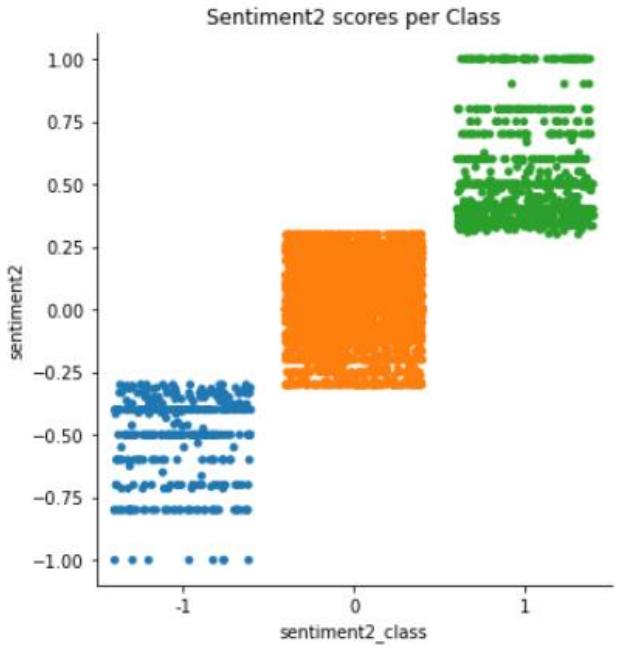


Figure 166 Display sentiment score after preprocessing

```
In [12]: # Paied t-test with spicy library

         # Generate 2 independent samples
         data1 = df['sentiment'].values.tolist()
         data2 = df['sentiment2'].values.tolist()

         stat, p = ttest_rel(data1, data2)
         print('Statistics=%.3f, p=%.3f' % (stat,p))

         Statistics=10.599, p=0.000
```

Figure 167 Output t-test statistics

```
In [13]: # Welch's non parameteric t-test
         data_tweets = df['sentiment2'].values.tolist()

         print('Variance sentiment2 : ', round(statistics.stdev(data_tweets), 3))

         Variance sentiment2 :  0.203
```

Figure 168 Output Welch' non parametric t-test result

```
In [15]: df.to_csv(r'C:\Users\weihe\Documents\RedditData\reddit_sentiment.csv', index = False, header = True)
```

Figure 169 Store processed data

5.3 Analysis on Stock market impact by Covid-19

In this section, I processed all the market date with sentiment data and store them into 1 CSV file then applied the SPSS to run the regression model.

5.3.1 Process of the data from stock market with sentiment collect from Twitter and Reddit

I loaded the library and data, before transformation is done as Figure 172

```
In [1]: import pandas as pd
        import numpy as np
```

Figure 170 Load library

```
In [2]: # process twitter data
        twitter = pd.read_csv("twitter_sentiment.csv")
        twitter.head()
```

			21T23:57:44.000Z	reports first I...				transmitted c...		'tran...	'trans.
1	1231005062165458945	en	2020-02-21T23:57:32.000Z	b'#Coronavirus quarantine in a Hubu Village, S...	1202960464549838848	1	2	quarantine Hubu Village Shuangpu Town Xihu D...	0.000000	['quarantine', 'Hubu', 'Village', 'Shuangpu', ...	['quarantine 'Hubu', 'Village 'Shuangpu', .
2	1231004398811275264	en	2020-02-21T23:54:54.000Z	b'Italy. I love this country. #coronavirusital...	1221955502340558848	0	3	Italy love country	0.500000	['Italy', 'love', 'country']	['Italy', 'love 'country
3	1230866203947798530	en	2020-02-21T23:53:24.000Z	b'@JoannaB36464141 @nicolamlow @alexvespi @Ada...	1093695069176000512	0	2	dispersion globallyn Analysis travel Wuhan av...	0.400000	['dispersion', 'globallyn', 'Analysis', 'trave...	['dispersior 'globallyr 'Analysis 'trave.
4	1231003954806501376	en	2020-02-21T23:53:08.000Z	b'@WHO \\nThis is a #pandemic \\n#CoronavirusO...	723319704710844417	0	0	This	0.000000	['This']	['This

Figure 171 Load data

```
In [3]: from datetime import datetime, timedelta
        #twitter['Date'] = datetime.datetime.strptime(twitter['timestamp'],"%Y-%m-%dT%H:%M:%S.%fZ")

        twitter['my_timestamp'] = pd.to_datetime(twitter['timestamp'], format='%Y-%m-%dT%H:%M:%S.%fZ')
        twitter['Date'] = [d.date() for d in twitter['my_timestamp']]
        twitter['Time'] = [d.time() for d in twitter['my_timestamp']]

In [4]: twitter['Date'] = pd.to_datetime(twitter['Date'], format='%Y-%m-%d')
```

Figure 172 Transform the Twitter Date

I created the average, variance and count for twitter and reddit data as Figure 173 to Figure 179

```
In [5]: twitter_grouped = twitter.groupby('Date').agg(
            {
                "sentiment2": ['mean', 'var', 'count']
            }
        )
```

```
In [6]: twitter_grouped.columns = twitter_grouped.columns.droplevel(level=0)
```

```
In [7]: twitter_grouped.head()
```

Out[7]:

	mean	var	count
Date			
2020-02-04	0.000000	NaN	1
2020-02-05	0.136364	NaN	1
2020-02-11	0.052372	0.047294	295
2020-02-12	0.042379	0.046027	502
2020-02-13	0.046617	0.041157	586

Figure 173 Group the daily statistics for twitter

```
In [9]: twitter_grouped.dropna(subset = ["var"], inplace=True)
```

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

Out[10]:

	mean	var	count
Date			
2020-02-11	0.052372	0.047294	295
2020-02-12	0.042379	0.046027	502
2020-02-13	0.046617	0.041157	586
2020-02-14	0.053361	0.051551	648
2020-02-15	0.038073	0.037860	536

Figure 174 Remove the null value from twitter

```
In [11]: # process reddit data
         reddit = pd.read_csv("reddit_sentiment.csv")
         reddit[['date','time']] = reddit.timestamp.str.split(" ",expand=True,)
         reddit['Date'] = pd.to_datetime(reddit['date'], format='%Y-%m-%d')
```

```
In [12]: reddit[['id','Date']].head()
```

Out[12]:

	id	Date
0	f2kpf6	2020-02-12
1	f2m2cv	2020-02-12
2	f2p51b	2020-02-12
3	f2qh4a	2020-02-12
4	f2roo5	2020-02-12

Figure 175 Load Reddit data and transform the data field

```
In [14]: reddit_grouped = reddit.groupby('Date').agg(
             {
                 "sentiment2": ['mean', 'var', 'count']
             }
         )
```

Figure 176 Aggregate the reddit records daily

```
In [15]: reddit_grouped.columns = reddit_grouped.columns.droplevel(level=0)
         reddit_grouped.head()
```

Out[15]:

	mean	var	count
Date			
2020-01-22	0.000000	NaN	1
2020-01-23	0.002841	0.011396	4
2020-01-24	0.100000	0.050000	5
2020-01-25	0.067160	0.041656	17
2020-01-26	0.062637	0.032373	26

Figure 177 Group the reddit records daily

```
In [15]: reddit_grouped.columns = reddit_grouped.columns.droplevel(level=0)
         reddit_grouped.head()
```

Out[15]:

Date	mean	var	count
2020-01-22	0.000000	NaN	1
2020-01-23	0.002841	0.011396	4
2020-01-24	0.100000	0.050000	5
2020-01-25	0.067160	0.041656	17
2020-01-26	0.062637	0.032373	26

Figure 178 Drop the level for the reddit group

```
In [17]: reddit_grouped.dropna(subset = ["var"], inplace=True)
```

Figure 179 Drop the row with null entries

I loaded the stock market data for SP500, VIX and FSI, then merged them together as Figure 180 to Figure 186

```
In [19]: #Load the S&P data
         #Load VIX data
         #Load FSI data

         sp = pd.read_csv("Download_INDEX_US_SP_US_SPX.csv",thousands=',' )
         vix = pd.read_csv("Download_INDEX_US_CBSX_VIX.csv")
         fsi = pd.read_csv("Download_fsi.csv")
```

Figure 180 Load the SP500 VIX and FSI stock market dataset

```
In [22]: # Add the Date field with correct data format
         sp['Date'] = pd.to_datetime(sp['Date'], format='%m/%d/%Y')
         vix['Date'] = pd.to_datetime(vix['Date'], format='%m/%d/%Y')
         fsi['Date'] = pd.to_datetime(fsi['Date'], format='%d/%m/%Y')
```

Figure 181 Update the Date field with valid format

```
In [25]: # Convert the field to float type
         sp['Close'] = sp['Close'].astype(float)
```

Figure 182 Update the Close fields to float type

```
In [26]: # merge S&P and VIX data
         df = pd.merge(sp[['Date','Close']],vix[['Date','Close']],on='Date', how='outer')
```

Figure 183 Merge SP with VIX based on Date

```
In [29]: # Merge in the OFR data
         merge1 = pd.merge(df[['Date','Close_x', 'Close_y']],fsi[['Date','OFR FSI']],on='Date', how='outer')
```

Figure 184 Merge with the FSI data

```
In [32]: # update the colume name with meaningful name
         merge1=merge1.rename(columns={
             "Close_x": "sp_index", "Close_y": "vix_index", "OFR FSI": "fsi"
         })
```

Figure 185 Rename the column

```
In [33]: merge1.head()
```

Out[33]:

	Date	sp_index	vix_index	fsi
0	2020-06-22	3117.86	31.77	-0.077
1	2020-06-19	3097.74	35.12	-0.005
2	2020-06-18	3115.34	32.94	0.110
3	2020-06-17	3113.49	33.47	0.040
4	2020-06-16	3124.74	33.67	0.123

Figure 186 Show the data

I merged in the Reddit and Twitter data to the stock market data onto the same dataframe.

```
In [34]: # merge with Reddit data
         merge2 = pd.merge(merge1,reddit_grouped,on='Date', how='outer')
```

```
In [35]: # update naming
         merge2.rename(columns={
             "mean": "reddit_mean", "var": "reddit_var", "count": "reddit_count"
         })
         merge2.head()
         print('Before:', merge2.columns)
         merge2.columns = ['Date', 'sp_index', 'vix_index', 'fsi', 'reddit_mean', 'reddit_var', 'reddit_count']
         print('After:', merge2.columns)

         Before: Index(['Date', 'sp_index', 'vix_index', 'fsi', 'mean', 'var', 'count'], dtype='object')
         After: Index(['Date', 'sp_index', 'vix_index', 'fsi', 'reddit_mean', 'reddit_var',
                'reddit_count'],
               dtype='object')
```

Figure 187 Merge with Reddit data

```
In [38]: # Merge in twitter data
         merge3 = pd.merge(merge2,twitter_grouped,on='Date', how='outer')
```

```
In [40]: # Rename colume
         print('Before:', merge3.columns)
         merge3.columns = ['Date', 'sp_index', 'vix_index', 'fsi', 'reddit_mean', 'reddit_var', 'reddit_count',  'twitter_mean', '
         print('After:', merge3.columns)

         Before: Index(['Date', 'sp_index', 'vix_index', 'fsi', 'reddit_mean', 'reddit_var',
                'reddit_count', 'mean', 'var', 'count'],
               dtype='object')
         After: Index(['Date', 'sp_index', 'vix_index', 'fsi', 'reddit_mean', 'reddit_var',
                'reddit_count', 'twitter_mean', 'twitter_var', 'twitter_count'],
               dtype='object')
```

Figure 188 Merge in the Twitter data

```
In [42]: merge3 = merge3.dropna()
```

```
In [43]: # Output the csv file for further processing
         merge3.to_csv(r'pca_dataset.csv', index = False, header = True)
```

```
In [44]: merge3.head()
```

Out[44]:

	Date	sp_index	vix_index	fsi	reddit_mean	reddit_var	reddit_count	twitter_mean	twitter_var	twitter_count
1	2020-06-19	3097.74	35.12	-0.005	0.055193	0.023722	31.0	0.073997	0.058878	8940.0
2	2020-06-18	3115.34	32.94	0.110	0.042210	0.039556	34.0	0.082638	0.062229	9522.0
3	2020-06-17	3113.49	33.47	0.040	0.049471	0.070616	35.0	0.075629	0.056596	9879.0
4	2020-06-16	3124.74	33.67	0.123	0.050189	0.036591	50.0	0.071805	0.057449	10064.0
5	2020-06-15	3066.59	34.40	0.700	0.142083	0.074444	37.0	0.075545	0.052938	8846.0

Figure 189 Drop the row with null values and output to CSV file

5.3.2 Analyze the stock market data with Covid-19 impact

I used the SPSS to load up the CSV file as Figure 190 to Figure 192

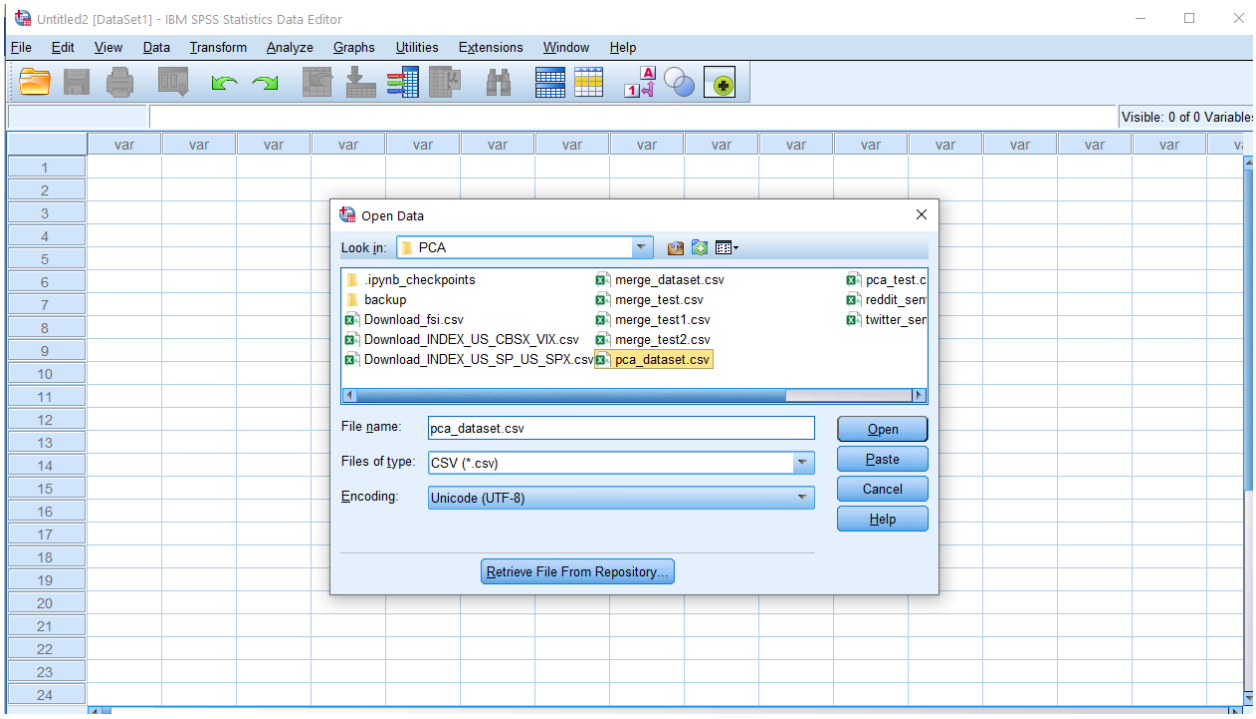
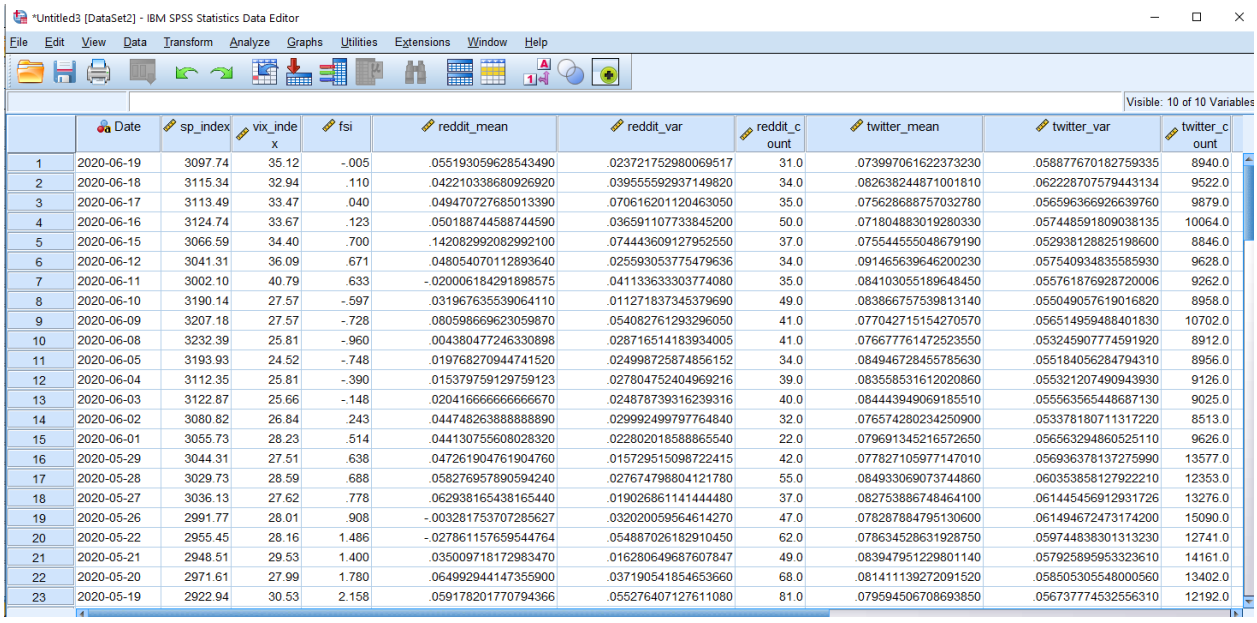
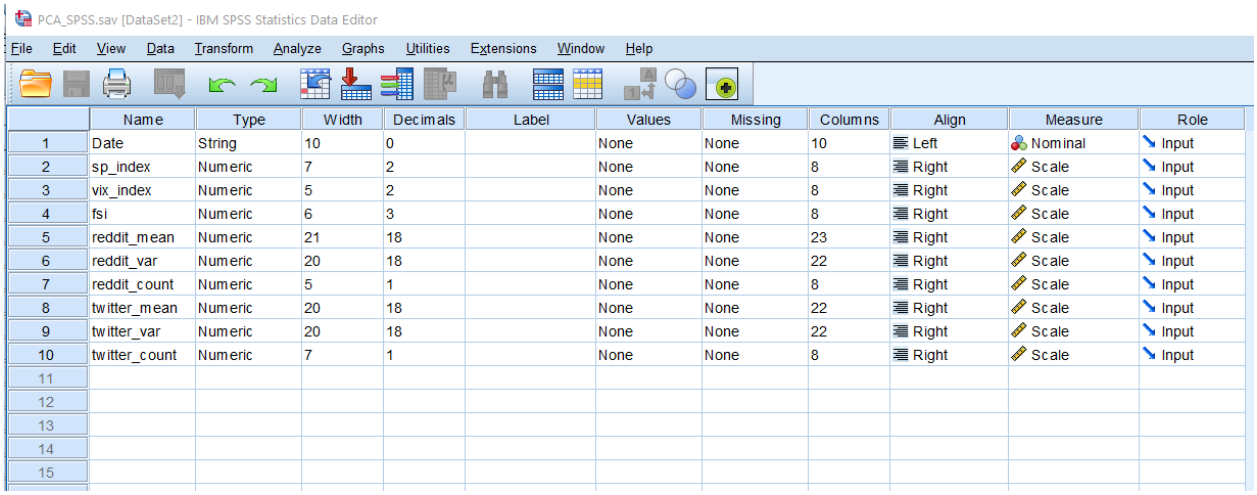


Figure 190 Load the CSV file



	Date	sp_index	vix_index	fsi	reddit_mean	reddit_var	reddit_count	twitter_mean	twitter_var	twitter_count
1	2020-06-19	3097.74	35.12	-.005	.055193059628543490	.023721752980069517	31.0	.073997061622373230	.058877670182759335	8940.0
2	2020-06-18	3115.34	32.94	.110	.042210338680926920	.039555592937149820	34.0	.082638244871001810	.062228707579443134	9522.0
3	2020-06-17	3113.49	33.47	.040	.049470727685013390	.070616201120463050	35.0	.075628688757032780	.056596366926639760	9879.0
4	2020-06-16	3124.74	33.67	.123	.050188744588744590	.036591107733845200	50.0	.071804883019280330	.057448591809038135	10064.0
5	2020-06-15	3066.59	34.40	.700	.142082992082992100	.074443609127952550	37.0	.075544555048679190	.052938128825198600	8846.0
6	2020-06-12	3041.31	36.09	.671	.048054070112893640	.025593053775479636	34.0	.091465639646200230	.057540934835585930	9628.0
7	2020-06-11	3002.10	40.79	.633	-.020006184291898575	.041133633303774080	35.0	.084103055189648450	.055761876928720006	9262.0
8	2020-06-10	3190.14	27.57	-.597	.031967635539064110	.011271837345379690	49.0	.083866757539813140	.055049057619016820	8958.0
9	2020-06-09	3207.18	27.57	-.728	.080598669623059870	.054082761293296050	41.0	.077042715154270570	.056514959488401830	10702.0
10	2020-06-08	3232.39	25.81	-.960	.004380477246330898	.028716514183934005	41.0	.076677761472523550	.053245907774591920	8912.0
11	2020-06-05	3193.93	24.52	-.748	.019768270944741520	.024998725874856152	34.0	.084946728455785630	.055184056284794310	8956.0
12	2020-06-04	3112.35	25.81	-.390	.015379759129759123	.027804752404969216	39.0	.083558531612020860	.055321207490943930	9126.0
13	2020-06-03	3122.87	25.66	-.148	.020416666666666670	.024878739316239316	40.0	.084443949069185510	.055563565448687130	9025.0
14	2020-06-02	3080.82	26.84	.243	.044748263888888890	.029992499797764840	32.0	.076574280234250900	.053378180711317220	8513.0
15	2020-06-01	3055.73	28.23	.514	.044130755608028320	.022802018588865540	22.0	.079691345216572650	.056563294860525110	9626.0
16	2020-05-29	3044.31	27.51	.638	.047261904761904760	.015729515098722415	42.0	.077827105977147010	.056936378137275990	13577.0
17	2020-05-28	3029.73	28.59	.688	.058276957890594240	.027674798804121780	55.0	.084933069073744860	.060353858127922210	12353.0
18	2020-05-27	3036.13	27.62	.778	.062938165438165440	.019026861141444480	37.0	.082753886748464100	.061445456912931726	13276.0
19	2020-05-26	2991.77	28.01	.908	-.003281753707285627	.032020059564614270	47.0	.078287884795130600	.061494672473174200	15090.0
20	2020-05-22	2955.45	28.16	1.486	-.027861157659544764	.054887026182910450	62.0	.078634528631928750	.059744838301313230	12741.0
21	2020-05-21	2948.51	29.53	1.400	.035009718172983470	.016280649687607847	49.0	.083947951229801140	.057925895953323610	14161.0
22	2020-05-20	2971.61	27.99	1.780	.064992944147355900	.037190541854653660	68.0	.081411139272091520	.058505305548000560	13402.0
23	2020-05-19	2922.94	30.53	2.158	.059178201770794366	.055276407127611080	81.0	.079594506708693850	.056737774532556310	12192.0

Figure 191 Display the data



	Name	Type	Width	Decimals	Label	Values	Missing	Columns	Align	Measure	Role
1	Date	String	10	0		None	None	10	Left	Nominal	Input
2	sp_index	Numeric	7	2		None	None	8	Right	Scale	Input
3	vix_index	Numeric	5	2		None	None	8	Right	Scale	Input
4	fsi	Numeric	6	3		None	None	8	Right	Scale	Input
5	reddit_mean	Numeric	21	18		None	None	23	Right	Scale	Input
6	reddit_var	Numeric	20	18		None	None	22	Right	Scale	Input
7	reddit_count	Numeric	5	1		None	None	8	Right	Scale	Input
8	twitter_mean	Numeric	20	18		None	None	22	Right	Scale	Input
9	twitter_var	Numeric	20	18		None	None	22	Right	Scale	Input
10	twitter_count	Numeric	7	1		None	None	8	Right	Scale	Input
11											
12											
13											
14											
15											

Figure 192 Display the data fields

I selected the SP500 index as Dependent Variable and other sentiment data as dependent vaiable and perform the linear regression to get the correlation. Coefficient, model summary, ANOVA for the entire set of sentiment data as Figure 193 to Figure 197

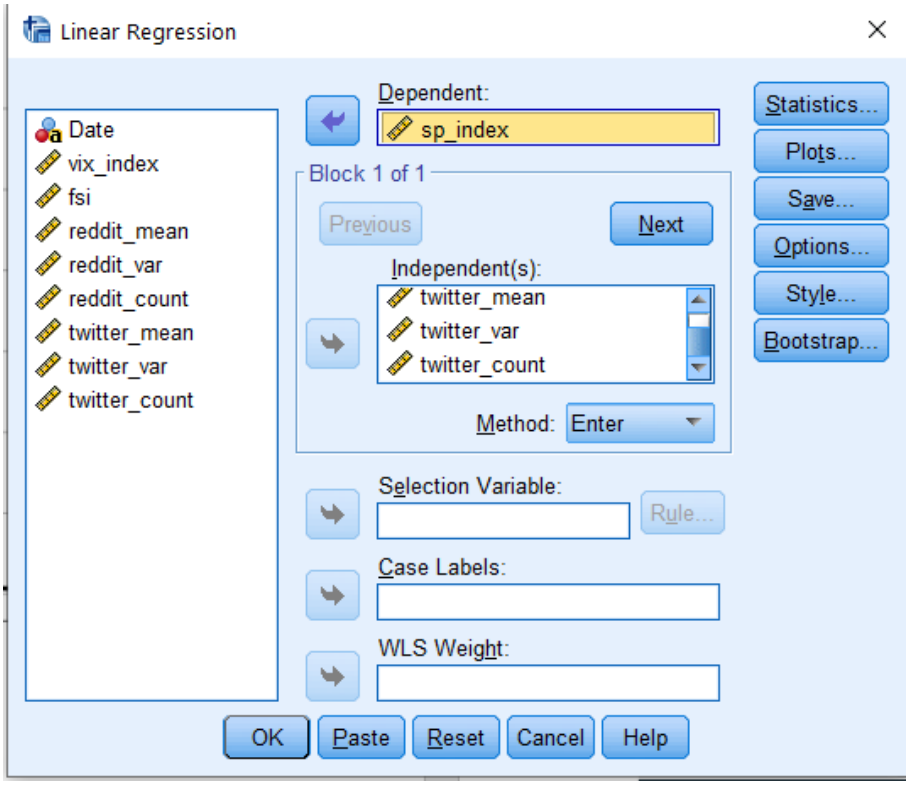


Figure 193 Define the IVs for DV as SP500 index

Correlations

		sp_index	reddit_mean	reddit_var	reddit_count	twitter_mean	twitter_var	twitter_count
Pearson Correlation	sp_index	1.000	-.107	-.102	-.833	-.551	-.593	-.801
	reddit_mean	-.107	1.000	.203	.029	.079	.030	.134
	reddit_var	-.102	.203	1.000	.150	-.134	-.168	.018
	reddit_count	-.833	.029	.150	1.000	.233	.375	.571
	twitter_mean	-.551	.079	-.134	.233	1.000	.724	.745
	twitter_var	-.593	.030	-.168	.375	.724	1.000	.666
	twitter_count	-.801	.134	.018	.571	.745	.666	1.000
Sig. (1-tailed)	sp_index	.	.157	.168	.000	.000	.000	.000
	reddit_mean	.157	.	.027	.394	.228	.390	.103
	reddit_var	.168	.027	.	.078	.103	.056	.433
	reddit_count	.000	.394	.078	.	.013	.000	.000
	twitter_mean	.000	.228	.103	.013	.	.000	.000
	twitter_var	.000	.390	.056	.000	.000	.	.000
	twitter_count	.000	.103	.433	.000	.000	.000	.
N	sp_index	91	91	91	91	91	91	91
	reddit_mean	91	91	91	91	91	91	91
	reddit_var	91	91	91	91	91	91	91
	reddit_count	91	91	91	91	91	91	91
	twitter_mean	91	91	91	91	91	91	91
	twitter_var	91	91	91	91	91	91	91
	twitter_count	91	91	91	91	91	91	91

Figure 194 Correlation with all stock data

Coefficients[a]

Model		Unstandardized Coefficients B	Unstandardized Coefficients Std. Error	Standardized Coefficients Beta	t	Sig.	95.0% Confidence Interval for B Lower Bound	95.0% Confidence Interval for B Upper Bound	Correlations Zero-order	Correlations Partial	Correlations Part	Collinearity Statistics Tolerance	Collinearity Statistics VIF
1	(Constant)	3744.599	163.259		22.937	.000	3419.941	4069.257					
	reddit_mean	-263.007	378.413	-.029	-.695	.489	-1015.523	489.508	-.107	-.076	-.028	.932	1.073
	reddit_var	-554.980	762.672	-.031	-.728	.469	-2071.637	961.676	-.102	-.079	-.029	.872	1.147
	reddit_count	-3.186	.291	-.589	-10.954	.000	-3.764	-2.607	-.833	-.767	-.440	.557	1.795
	twitter_mean	-2471.494	1443.668	-.126	-1.712	.091	-5342.387	399.399	-.551	-.184	-.069	.298	3.359
	twitter_var	-3896.837	3314.175	-.074	-1.176	.243	-10487.438	2693.763	-.593	-.127	-.047	.411	2.435
	twitter_count	-.011	.003	-.317	-4.088	.000	-.016	-.005	-.801	-.407	-.164	.268	3.733

a. Dependent Variable: sp_index

Figure 195 Coefficients with all stock data

Model Summary[b]

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate	Change Statistics R Square Change	Change Statistics F Change	Change Statistics df1	Change Statistics df2	Change Statistics Sig. F Change	Durbin-Watson
1	.930[a]	.865	.855	101.83597	.865	89.329	6	84	.000	1.542

a. Predictors: (Constant), twitter_count, reddit_var, reddit_mean, reddit_count, twitter_var, twitter_mean

b. Dependent Variable: sp_index

Figure 196 Model summary with all stock data

ANOVA[a]

Model		Sum of Squares	df	Mean Square	F	Sig.
1	Regression	5558365.260	6	926394.210	89.329	.000[b]
	Residual	871127.404	84	10370.564		
	Total	6429492.664	90			

a. Dependent Variable: sp_index

b. Predictors: (Constant), twitter_count, reddit_var, reddit_mean, reddit_count, twitter_var, twitter_mean

Figure 197 ANOVA with all stock data

The above result showed that the not all the IVs are significant. I selected with SP_index and control the Redit_count, Twitter_mean, Twitter_var, Twitter _count as independent variable as Figure 198 to Figure 203

I removed the Reddit_mean and Reddit_var from the independent variable, re-run the model and it show all remaining 4 IVs are significant. Adjusted R Square has reduced from 0.904 to 0.902 which still maintained at very high level, which indicates the validity of the model. The Coefficients shows the significant predictors of the variable of Reddit_count and Twitter_count are under 0.05 which indicates these 2 variables contribute more to the model to predict the S&P500 movement. Pearson correlation results support the result from the linear regression.

Correlations

		sp_index	reddit_count	twitter_mean	twitter_var	twitter_count
Pearson Correlation	sp_index	1.000	-.833	-.551	-.593	-.801
	reddit_count	-.833	1.000	.233	.375	.571
	twitter_mean	-.551	.233	1.000	.724	.745
	twitter_var	-.593	.375	.724	1.000	.666
	twitter_count	-.801	.571	.745	.666	1.000
Sig. (1-tailed)	sp_index	.	.000	.000	.000	.000
	reddit_count	.000	.	.013	.000	.000
	twitter_mean	.000	.013	.	.000	.000
	twitter_var	.000	.000	.000	.	.000
	twitter_count	.000	.000	.000	.000	.
N	sp_index	91	91	91	91	91
	reddit_count	91	91	91	91	91
	twitter_mean	91	91	91	91	91
	twitter_var	91	91	91	91	91
	twitter_count	91	91	91	91	91

Figure 198 Correlation with new set of variables

Model Summary[b]

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate	Change Statistics					Durbin-Watson
					R Square Change	F Change	df1	df2	Sig. F Change	
1	.929[a]	.862	.856	101.40237	.862	134.822	4	86	.000	1.518

a. Predictors: (Constant), twitter_count, reddit_count, twitter_var, twitter_mean

b. Dependent Variable: sp_index

Figure 199 Model summary with new set of variables

Coefficients[a]

Model		Unstandardized Coefficients		Standardized Coefficients	t	Sig.	95.0% Confidence Interval for B		Correlations			Collinearity Statistics	
		B	Std. Error	Beta			Lower Bound	Upper Bound	Zero-order	Partial	Part	Tolerance	VIF
1	(Constant)	3677.580	148.493		24.766	.000	3382.386	3972.774					
	reddit_count	-3.202	.286	-.592	-11.206	.000	-3.770	-2.634	-.833	-.770	-.448	.572	1.748
	twitter_mean	-2381.969	1434.347	-.121	-1.661	.100	-5233.356	469.418	-.551	-.176	-.066	.299	3.344
	twitter_var	-3297.988	3242.367	-.062	-1.017	.312	-9743.599	3147.624	-.593	-.109	-.041	.425	2.351
	twitter_count	-.011	.003	-.331	-4.338	.000	-.016	-.006	-.801	-.424	-.173	.275	3.638

a. Dependent Variable: sp_index

Figure 200 Coefficients with new set of variables

I ran the linear regression again with the VIX as dependent variable, I got statistics result as Figure 201 to Figure 205

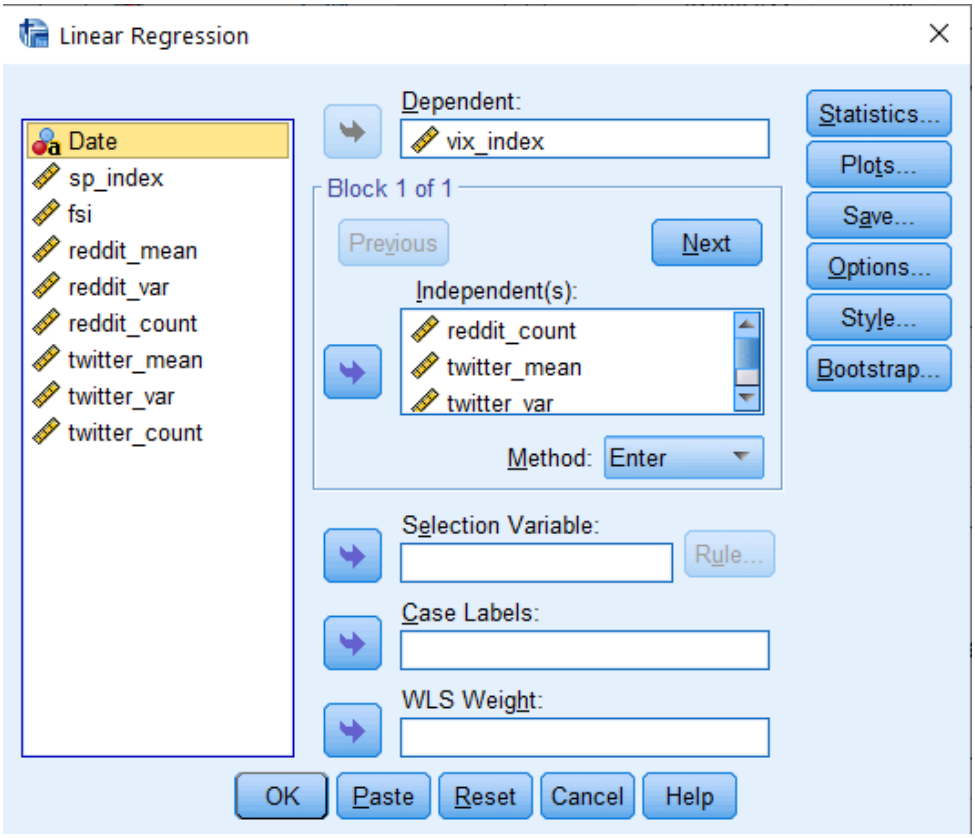


Figure 201 Define the IVs for DV as VIX index

Correlations

		vix_index	reddit_mean	reddit_var	reddit_count	twitter_mean	twitter_var	twitter_count
Pearson Correlation	vix_index	1.000	.056	.149	.884	.374	.454	.633
	reddit_mean	.056	1.000	.203	.029	.079	.030	.134
	reddit_var	.149	.203	1.000	.150	-.134	-.168	.018
	reddit_count	.884	.029	.150	1.000	.233	.375	.571
	twitter_mean	.374	.079	-.134	.233	1.000	.724	.745
	twitter_var	.454	.030	-.168	.375	.724	1.000	.666
	twitter_count	.633	.134	.018	.571	.745	.666	1.000
Sig. (1-tailed)	vix_index	.	.298	.080	.000	.000	.000	.000
	reddit_mean	.298	.	.027	.394	.228	.390	.103
	reddit_var	.080	.027	.	.078	.103	.056	.433
	reddit_count	.000	.394	.078	.	.013	.000	.000
	twitter_mean	.000	.228	.103	.013	.	.000	.000
	twitter_var	.000	.390	.056	.000	.000	.	.000
	twitter_count	.000	.103	.433	.000	.000	.000	.
N	vix_index	91	91	91	91	91	91	91
	reddit_mean	91	91	91	91	91	91	91
	reddit_var	91	91	91	91	91	91	91
	reddit_count	91	91	91	91	91	91	91
	twitter_mean	91	91	91	91	91	91	91
	twitter_var	91	91	91	91	91	91	91
	twitter_count	91	91	91	91	91	91	91

Figure 202 Correlation with all stock data

Coefficients[a]

Model		Unstandardized Coefficients B	Unstandardized Coefficients Std. Error	Standardized Coefficients Beta	t	Sig.	95.0% Confidence Interval for B Lower Bound	95.0% Confidence Interval for B Upper Bound	Correlations Zero-order	Correlations Partial	Correlations Part	Collinearity Statistics Tolerance	Collinearity Statistics VIF
1	(Constant)	-1.426	10.937		-.130	.897	-23.176	20.323					
	reddit_mean	2.345	25.351	.005	.092	.927	-48.068	52.757	.056	.010	.004	.932	1.073
	reddit_var	49.360	51.093	.049	.966	.337	-52.243	150.964	.149	.105	.045	.872	1.147
	reddit_count	.248	.019	.802	12.737	.000	.209	.287	.884	.812	.598	.557	1.795
	twitter_mean	143.170	96.714	.127	1.480	.143	-49.156	335.497	.374	.159	.070	.298	3.359
	twitter_var	90.815	222.023	.030	.409	.684	-350.701	532.331	.454	.045	.019	.411	2.435
	twitter_count	.000	.000	.059	.649	.518	.000	.000	.633	.071	.031	.268	3.733

a. Dependent Variable: vix_index

Figure 203 Coefficients with all variables

Model Summary[b]

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate	Change Statistics R Square Change	Change Statistics F Change	Change Statistics df1	Change Statistics df2	Change Statistics Sig. F Change	Durbin-Watson
1	.903[a]	.815	.801	6.82217	.815	61.501	6	84	.000	1.191

a. Predictors: (Constant), twitter_count, reddit_var, reddit_mean, reddit_count, twitter_var, twitter_mean

b. Dependent Variable: vix_index

Figure 204 Model summary with all variables

ANOVA[a]

Model		Sum of Squares	df	Mean Square	F	Sig.
1	Regression	17174.411	6	2862.402	61.501	.000[b]
	Residual	3909.532	84	46.542		
	Total	21083.944	90			

a. Dependent Variable: vix_index

b. Predictors: (Constant), twitter_count, reddit_var, reddit_mean, reddit_count, twitter_var, twitter_mean

Figure 205 ANOVA with all variables

I removed the Reddit_mean and reddit_var from the independent variable, re-run the model and it show all remaining 4 IVs are significant. The rerun result is shown in Figure 206 to Figure 209. Adjusted R Square has reduced from 0.904 to 0.902 which still maintained at very high level, which indicates the validity of the model. The Coefficients shows the significant predictors of the variable of twitter_count is under 0.05 which indicates this variable contribute more to the model to predict the VIX movement. Pearson correlation results also support the result from the linear regression.

Correlations

		vix_index	reddit_count	twitter_mean	twitter_var	twitter_count
Pearson Correlation	vix_index	1.000	.884	.374	.454	.633
	reddit_count	.884	1.000	.233	.375	.571
	twitter_mean	.374	.233	1.000	.724	.745
	twitter_var	.454	.375	.724	1.000	.666
	twitter_count	.633	.571	.745	.666	1.000
Sig. (1-tailed)	vix_index	.	.000	.000	.000	.000
	reddit_count	.000	.	.013	.000	.000
	twitter_mean	.000	.013	.	.000	.000
	twitter_var	.000	.000	.000	.	.000
	twitter_count	.000	.000	.000	.000	.
N	vix_index	91	91	91	91	91
	reddit_count	91	91	91	91	91
	twitter_mean	91	91	91	91	91
	twitter_var	91	91	91	91	91
	twitter_count	91	91	91	91	91

Figure 206 Correlation with subset of IVs

Model Summary[b]

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate	Change Statistics					Durbin-Watson
					R Square Change	F Change	df1	df2	Sig. F Change	
1	.901[a]	.812	.804	6.78318	.812	93.058	4	86	.000	1.177

a. Predictors: (Constant), twitter_count, reddit_count, twitter_var, twitter_mean

b. Dependent Variable: vix_index

Figure 207 Model Summary with subset of IVs

Coefficients[a]

Model		Unstandardized Coefficients		Standardized Coefficients	t	Sig.	95.0% Confidence Interval for B		Correlations			Collinearity Statistics	
		B	Std. Error	Beta			Lower Bound	Upper Bound	Zero-order	Partial	Part	Tolerance	VIF
1	(Constant)	3.041	9.933		.306	.760	-16.706	22.787					
	reddit_count	.251	.019	.810	13.111	.000	.213	.289	.884	.816	.612	.572	1.748
	twitter_mean	136.684	95.949	.122	1.425	.158	-54.056	327.424	.374	.152	.067	.299	3.344
	twitter_var	49.107	216.894	.016	.226	.821	-382.064	480.278	.454	.024	.011	.425	2.351
	twitter_count	.000	.000	.069	.778	.439	.000	.000	.633	.084	.036	.275	3.638

a. Dependent Variable: vix_index

Figure 208 Coefficients with subset of IVs

I ran the linear regression again with the FSI as dependent variable, I got statistics result as Figure 210 to Figure 213

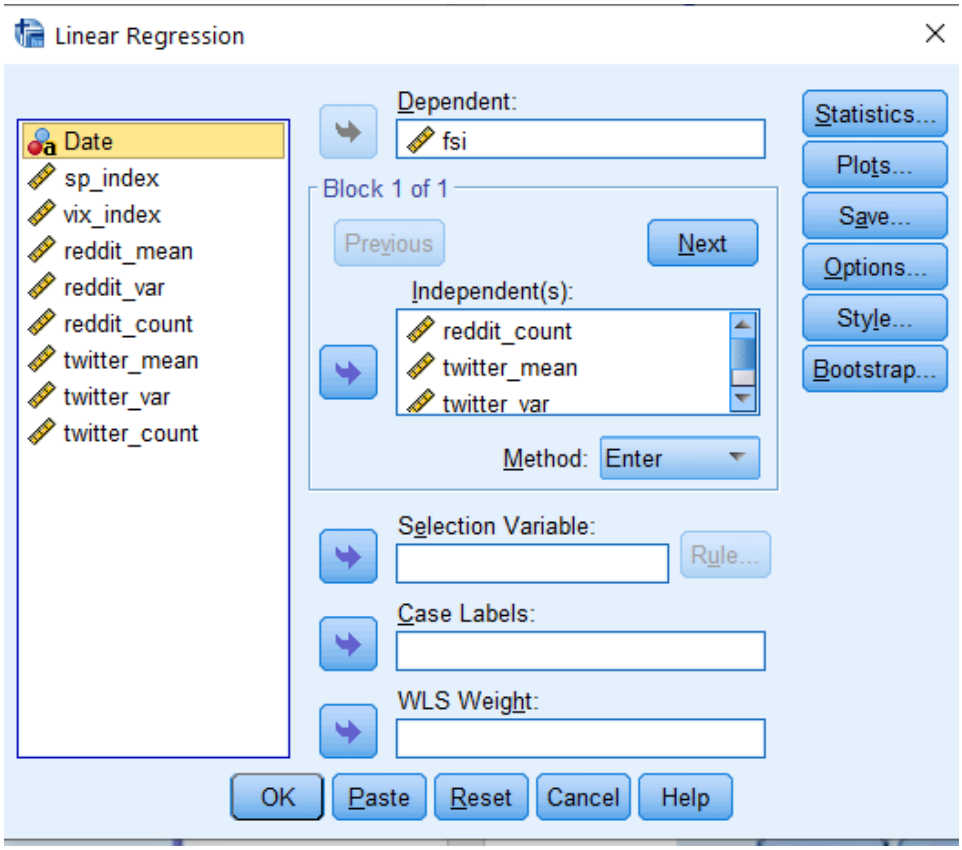


Figure 209 Define the IVs for DV as FSI index

Correlations

		fsi	reddit_mean	reddit_var	reddit_count	twitter_mean	twitter_var	twitter_count
Pearson Correlation	fsi	1.000	.126	.110	.821	.607	.618	.855
	reddit_mean	.126	1.000	.203	.029	.079	.030	.134
	reddit_var	.110	.203	1.000	.150	-.134	-.168	.018
	reddit_count	.821	.029	.150	1.000	.233	.375	.571
	twitter_mean	.607	.079	-.134	.233	1.000	.724	.745
	twitter_var	.618	.030	-.168	.375	.724	1.000	.666
	twitter_count	.855	.134	.018	.571	.745	.666	1.000
Sig. (1-tailed)	fsi	.	.117	.150	.000	.000	.000	.000
	reddit_mean	.117	.	.027	.394	.228	.390	.103
	reddit_var	.150	.027	.	.078	.103	.056	.433
	reddit_count	.000	.394	.078	.	.013	.000	.000
	twitter_mean	.000	.228	.103	.013	.	.000	.000
	twitter_var	.000	.390	.056	.000	.000	.	.000
	twitter_count	.000	.103	.433	.000	.000	.000	.
N	fsi	91	91	91	91	91	91	91
	reddit_mean	91	91	91	91	91	91	91
	reddit_var	91	91	91	91	91	91	91
	reddit_count	91	91	91	91	91	91	91
	twitter_mean	91	91	91	91	91	91	91
	twitter_var	91	91	91	91	91	91	91
	twitter_count	91	91	91	91	91	91	91

Figure 210 Correlation with all stock data

Coefficients[a]

Model		Unstandardized Coefficients B	Unstandardized Coefficients Std. Error	Standardized Coefficients Beta	t	Sig.	95.0% Confidence Interval for B Lower Bound	95.0% Confidence Interval for B Upper Bound	Correlations Zero-order	Correlations Partial	Correlations Part	Collinearity Statistics Tolerance	Collinearity Statistics VIF
1	(Constant)	-8.729	1.854		-4.709	.000	-12.414	-5.043					
	reddit_mean	4.433	4.296	.035	1.032	.305	-4.111	12.976	.126	.112	.034	.932	1.073
	reddit_var	11.018	8.659	.045	1.272	.207	-6.201	28.237	.110	.138	.042	.872	1.147
	reddit_count	.040	.003	.532	12.128	.000	.033	.047	.821	.798	.397	.557	1.795
	twitter_mean	42.310	16.390	.155	2.581	.012	9.716	74.904	.607	.271	.084	.298	3.359
	twitter_var	34.925	37.627	.047	.928	.356	-39.900	109.751	.618	.101	.030	.411	2.435
	twitter_count	.000	.000	.399	6.309	.000	.000	.000	.855	.567	.206	.268	3.733

a. Dependent Variable: fsi

Figure 211 Coefficient with all stock data

Model Summary[b]

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate	Change Statistics R Square Change	Change Statistics F Change	Change Statistics df1	Change Statistics df2	Change Statistics Sig. F Change	Durbin-Watson
1	.954[a]	.910	.904	1.156177	.910	141.641	6	84	.000	1.553

a. Predictors: (Constant), twitter_count, reddit_var, reddit_mean, reddit_count, twitter_var, twitter_mean

b. Dependent Variable: fsi

Figure 212 Model Summary with all stock data

ANOVA[a]

Model		Sum of Squares	df	Mean Square	F	Sig.
1	Regression	1136.032	6	189.339	141.641	.000[b]
	Residual	112.287	84	1.337		
	Total	1248.319	90			

a. Dependent Variable: fsi

b. Predictors: (Constant), twitter_count, reddit_var, reddit_mean, reddit_count, twitter_var, twitter_mean

Figure 213 ANOVA with all stock data

Remove the Reddit_mean and Reddit_var from the independent variable, re-run the model and it show all remaining 4 IVs are significant. The result of new run is as Figure 214 to Figure 216. Adjusted R Square has reduced from 0.904 to 0.902 which still maintained at very high level, which indicates the validity of the model. The Coefficients shows the significant predictors of the variable of Reddit_count, Twitter_mean and Twitter_count were under 0.05 which indicates these 3 variables contribute more to the model. Pearson correlation results support the result from the linear regression.

Correlations

		fsi	reddit_count	twitter_mean	twitter_var	twitter_count
Pearson Correlation	fsi	1.000	.821	.607	.618	.855
	reddit_count	.821	1.000	.233	.375	.571
	twitter_mean	.607	.233	1.000	.724	.745
	twitter_var	.618	.375	.724	1.000	.666
	twitter_count	.855	.571	.745	.666	1.000
Sig. (1-tailed)	fsi	.	.000	.000	.000	.000
	reddit_count	.000	.	.013	.000	.000
	twitter_mean	.000	.013	.	.000	.000
	twitter_var	.000	.000	.000	.	.000
	twitter_count	.000	.000	.000	.000	.
N	fsi	91	91	91	91	91
	reddit_count	91	91	91	91	91
	twitter_mean	91	91	91	91	91
	twitter_var	91	91	91	91	91
	twitter_count	91	91	91	91	91

Figure 214 Correlations with subset of IVs

Model Summary[b]

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate	Change Statistics					Durbin-Watson
					R Square Change	F Change	df1	df2	Sig. F Change	
1	.952[a]	.906	.902	1.165231	.906	208.349	4	86	.000	1.449

a. Predictors: (Constant), twitter_count, reddit_count, twitter_var, twitter_mean

b. Dependent Variable: fsi

Figure 215 Model summary with subset of IVs

Coefficients[a]

Model		Unstandardized Coefficients		Standardized Coefficients	t	Sig.	95.0% Confidence Interval for B		Correlations			Collinearity Statistics	
		B	Std. Error	Beta			Lower Bound	Upper Bound	Zero-order	Partial	Part	Tolerance	VIF
1	(Constant)	-7.454	1.706		-4.368	.000	-10.846	-4.062					
	reddit_count	.040	.003	.537	12.307	.000	.034	.047	.821	.799	.406	.572	1.748
	twitter_mean	40.588	16.482	.149	2.463	.016	7.822	73.354	.607	.257	.081	.299	3.344
	twitter_var	23.470	37.259	.032	.630	.530	-50.598	97.537	.618	.068	.021	.425	2.351
	twitter_count	.000	.000	.417	6.623	.000	.000	.000	.855	.581	.218	.275	3.638

a. Dependent Variable: fsi

Figure 216 Coefficients with subset of IVs