IBM Hack Challenge 2020

SENTIMENT ANALYSIS OF COVID-19 TWEETS

By Team

Cassiopeia

Team Members:

Namitha S

Meghana R

Nikhitha Gururaj

TABLE OF CONTENTS

Link to presentation video :
https://drive.google.com/file/d/1QugLC7yCr5skR929sbqpnTNt_tclwiJe/view

Link to the website :
https://sentiment-analysis-dashboard-cassiopeia.000webhostapp.com/

INTRODUCTION

1.1 OVERVIEW

Sentiment Analysis is the process of predicting whether a piece of information indicates a positive, negative or neutral sentiment on the topic using natural language processing.

The dataset is obtained by querying the twitter API. Python library files are used for pre-processing of the extracted tweets to suitable data for further processing. In this stage the hashtags are obtained for analysis. NLTK-Naïve Bayes classifier and Vader technique are used to identify the sentiment of the people across the various stages of lockdown.

By visualizing data in various formats we are able to give a complete view of the sentiment of the people during the COVID19 pandemic.

1.2 PURPOSE

In the past two decades, the growth of social data on the web has rapidly increased. Social data on the web contains many real life events that occurred in daily life, today the global COVID-19 disease is spread worldwide. Many individuals including media organizations and government agencies are presenting the latest news and opinions regarding the coronavirus.

LITERATURE SURVEY

2.1 EXISTING PROBLEM

Sentiment analysis is a growing area of Natural Language Processing with research ranging from document level classification to learning the polarity of words and phrases. Given the character limitations on tweets, classifying the sentiment of Twitter messages is most similar to sentence level sentiment ;however, the informal and specialized language used in tweets, as well as the very nature of the microblogging domain make Twitter sentiment analysis a very different task.

2.2 PROPOSED SOLUTION

People express their sentiments and opinions through various means. Our analysis takes this into account by using the relevant hashtags to mine data. The data obtained is used to get the general public opinion regarding the pandemic among the Indian twitter users.

Data extraction:

The data is obtained from Kaggle(COVID-19 Twitter Dataset India) and by web scrapping using GetOldTweets3 API.

Data cleaning:

Transforms the raw data into valuable data. Real-world data has so much noise and is often incomplete, inconsistent, and lacking in certain behaviors or trends.

It involves removal of hyper links, mentions, numerical, punctuations and hashtags by using python preprocessor library and regular expressions.

Analysis using NLTK-Naïve Bayes:

NLTK consists of the most common algorithms such as tokenizing, part-of-speech tagging, stemming, sentiment analysis, topic segmentation, and named entity recognition. NLTK helps the computer to analysis, preprocess, and understand the written text.

Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

Analysis using Vader technique:

VADER stands for Valence Aware Dictionary and sentiment reasoning, is a lexicon and rule-based tool that is specifically tuned to social media. Given a string of text, it outputs a decimal between 0 and 1 for each of negativity, positivity, and neutrality for the text, as well as a compound score from -1 to 1 which is an aggregate measure.

The technique is used to identify the emotions like anger, anticipation, disgust, fear, joy, sadness, surprise and trust.

Visualization:

This analysis is further extended to visualize the reaction of the citizens towards initiatives like Jantha curfew , 9Baje9minutes and Aarogya Setu .The visualization also includes geographic

based visualization of state wise tweets, word cloud of frequently used hashtags, month wise emotion analysis of tweets and overall positive/negative count of tweets.

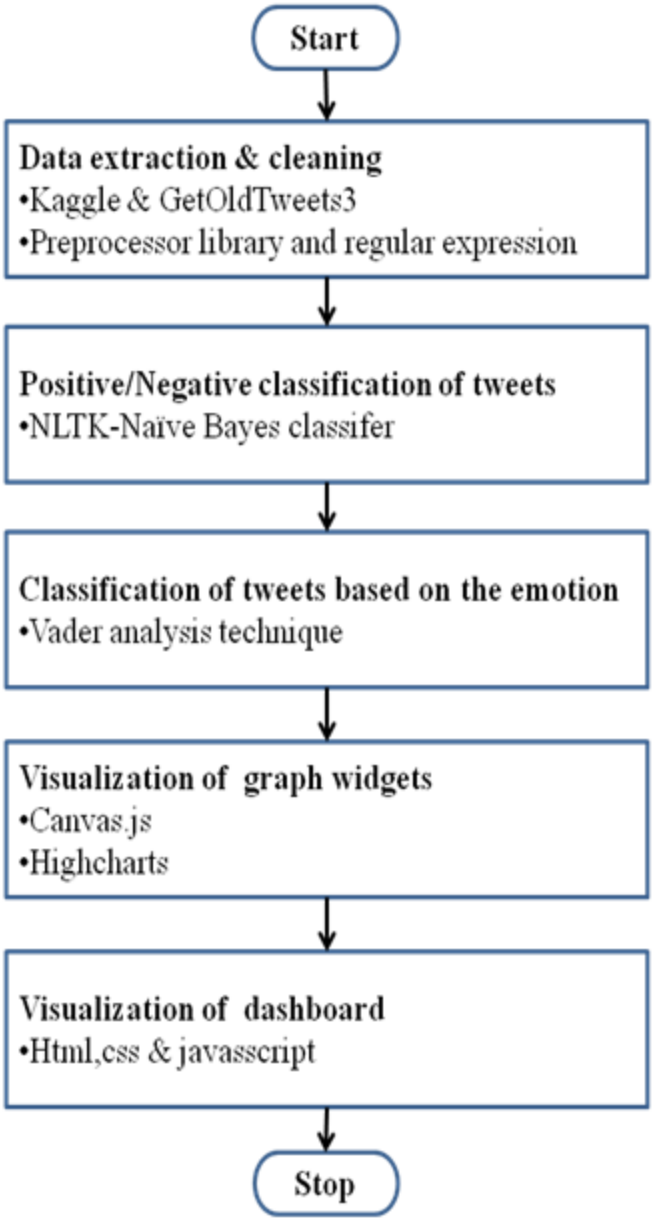
THEORITICAL ANALYSIS

3.1 BLOCK DIAGRAM

3.2 HARDWARE /SOFTWARE DESIGNING

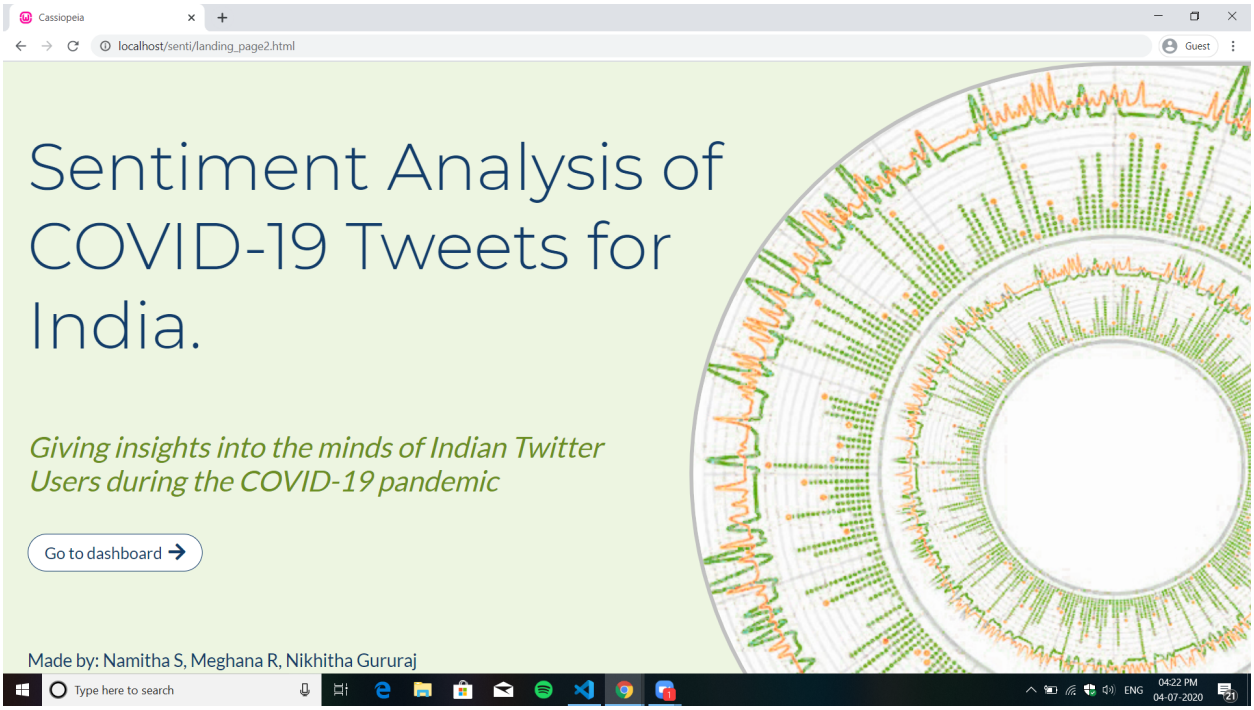
- Kaggle and GetOldTweets3 API is used to obtain the tweets.
- Python is used as the primary programming language.
- NLTK, is a suite of libraries and programs used for symbolic and statistical natural language processing (NLP).
- NRC-Sentiment-Emotion-Lexicons is used for analysis using Vader.
- Python libraries like NumPy, Pandas and preprocessor are used for analyzing the tweets. Visualization libraries like matplotlib and wordcloud are used.
- IBM Watson Studio helps to prepare data and build models at scale across any cloud.
- hosting
- Html, css and javascript are used for the creation of UI.

FLOWCHART

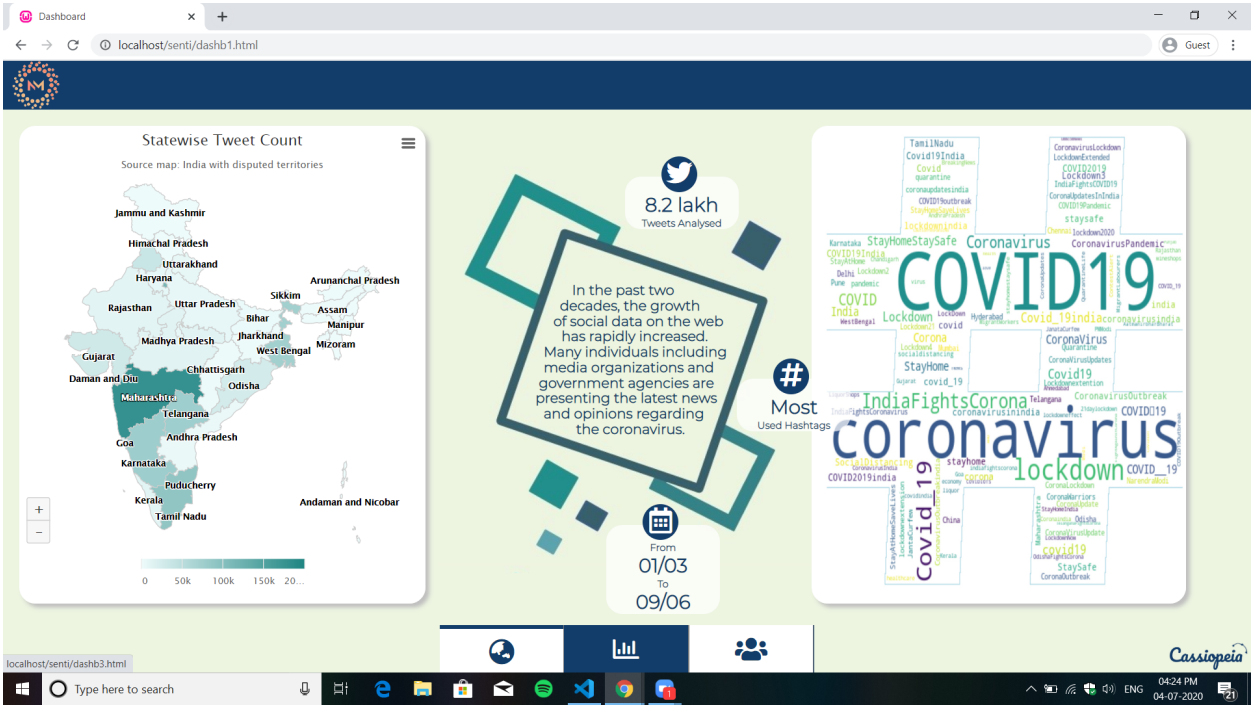

Start
Data extraction & cleaning
•Kaggle & GetOldTweets3
•Preprocessor library and regular expression
Positive/Negative classification of tweets
•NLTK-Naïve Bayes classifer
Classification of tweets based on the emotion
•Vader analysis technique
Visualization of graph widgets
•Canvas.js
•Highcharts
Visualization of dashboard
•Html,css & javasscript
Stop

RESULTS

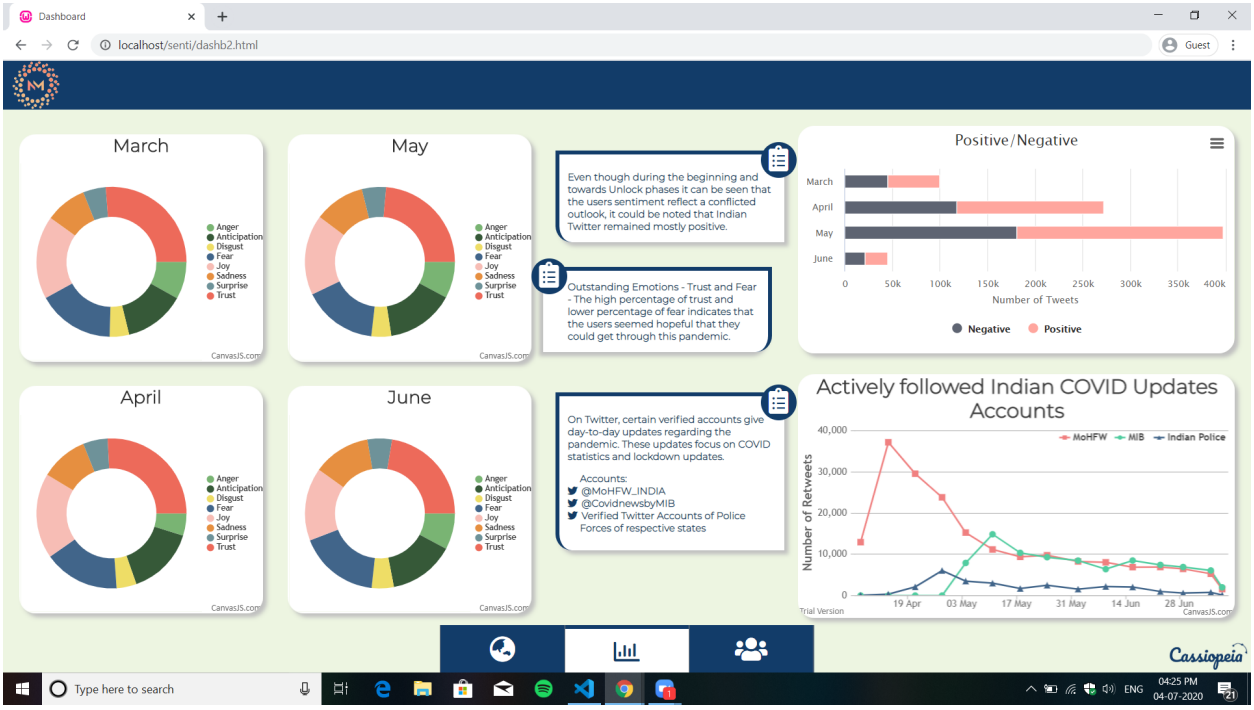
Dashboard Landing Page :



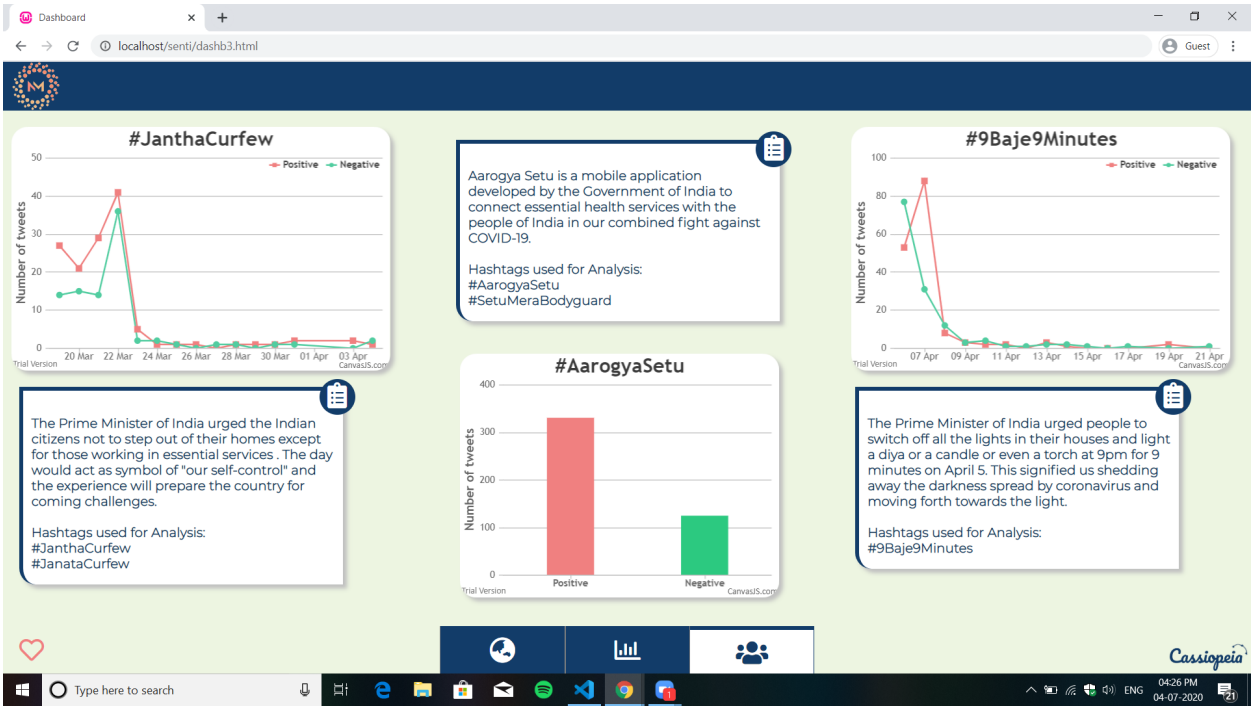
Page with geographic visualization and most frequently used hashtags:



Page consisting monthwise emotion analysis, overall positive/negative and visualization of actively followed Indian covid updates accounts:



Page with analysis of Indian Government Intiatives like Jantha Cerfew, 9Baje9Minutes and Arogya Setu:



ADVANTAGES & DISADVANTAGES

7.1 ADVANTAGES

- You can develop a more insightful, decision making strategy.
- This application is not only convenient for Coronavirus health issue but it can also be adopted as model to discover sentiment emotion for the future similar cases.
- The analysis takes into account the people's feelings will be helpful in identifying the society's problems and strengths.

7.2 DISADVANTAGES

- Recognizing things like sarcasm and irony, negations, jokes, and exaggerations can't be easily classified.
- It is not possible to analyze the large amount of twitter data without error.
- The results are dependent on the natural language processing techniques used.

APPLICATIONS

- The analysis of these tweets gives an overview of the acceptance and feedback of the people on the Government decisions which are helpful in taking future decision.
- The use of this information can be applied to make wiser decisions related to the use of resources, to make improvements in organizations, providing better services, and ultimately to improve the citizen lifestyle and the human relations in order to achieve a better society.

CONCLUSION

In this project, we analyzed the sentiments of COVID-19-related tweets in several ways. The overall trend shows that the public has been more optimistic over time. To fight the coronavirus not only needs the guidance from the government but also a positive attitude from the public.

Our analysis provides a potential approach to reveal the public's sentiment status and help institutions respond timely to it**.**

We addressed issues surrounding public sentiment reflecting deep concerns about Coronavirus and COVID-19, leading to the identification of various sentiments.

FUTURE SCOPE

Due to lack of time, and computational process, many aspects has been left for the future works. It would be interesting to take the following research area into consideration:

- Since there are so many professional and official people on Twitter, you may find more reliable source of information on Twitter than other social Medias such as Facebook, Instagram. However, it is very essential to explore other social media with regard to sentiment analysis.
- In addition to using existing dataset, live feed of tweets can be taken into account.
- Since India is a multi-linguistic nation regional languages can also be considered in addition to the English language.

BIBILOGRAPHY

- https://github.com/SivaAndMe/Sentiment-Analysis-on-Swachh-Bharat-using-twitter/blob/master/convert_text_to_csv.py
- https://towardsdatascience.com/extracting-twitter-data-pre-processing-and-sentiment-analysis-using-python-3-0-7192bd8b47cf
- https://stackoverflow.com/questions/2527892/parsing-a-tweet-to-extract-hashtags-into-an-array
- http://www.laurentluce.com/posts/twitter-sentiment-analysis-using-python-and-nltk/
- https://www.geeksforgeeks.org/twitter-sentiment-analysis-using-python/
- https://towardsdatascience.com/basic-nlp-on-the-texts-of-harry-potter-sentiment-analysis-1b474b13651d
- https://towardsdatascience.com/creating-the-twitter-sentiment-analysis-program-in-python-with-naive-bayes-classification-672e5589a7ed

- https://www.twilio.com/blog/2017/12/sentiment-analysis-scikit-learn.html
- https://sci-hub.tw/10.1109/aitc.2019.8921014
- https://medium.com/analytics-vidhya/using-nlp-to-determine-the-sentiments-of-tweets-522f1cca381a
- https://canvasjs.com/
- https://towardsdatascience.com/create-word-cloud-into-any-shape-you-want-using-python-d0b88834bc32
- https://www.datacamp.com/community/tutorials/wordcloud-python
- https://www.highcharts.com

APPENDIX

Collection of data and convert to csv

GetOldTweets3.py

```
1  import os, sys, re, getopt
2  import traceback
3
4  if sys.version_info[0] < 3:
5      raise Exception("Python 2.x is not supported. Please
   upgrade to 3.x")
6
7  import GetOldTweets3 as got
8
9  def main(argv):
10     if len(argv) == 0:
11         print('You must pass some parameters. Use \"-h\" to
   help.')
12         return
13
14     if len(argv) == 1 and argv[0] == '-h':
15         print(__doc__)
16         return
```

```
17
18      try:
19          opts, args = getopt.getopt(argv, "",
    ("querysearch=","username=","usernames-from-file=","since=",
    "until=","toptweets","maxtweets=","lang=","output="))
20
21
22          tweetCriteria = got.manager.TweetCriteria()
23          outputFileName = "output_got.csv"
24          debug = False
25          usernames = set()
26          username_files = set()
27          for opt, arg in opts:
28              if opt == '--querysearch':
29                  tweetCriteria.querySearch = arg
30              elif opt == '--username':
31                  usernames_ = [u.lstrip('@') for u in
    re.split(r'[\s,]+', arg) if u]
32                  usernames_ = [u.lower() for u in usernames_
    if u]
33                  usernames |= set(usernames_)
34              elif opt == '--usernames-from-file':
35                  username_files.add(arg)
36              elif opt == '--since':
37                  tweetCriteria.since = arg
38              elif opt == '--until':
39                  tweetCriteria.until = arg
40              elif opt == '--toptweets':
41                  tweetCriteria.topTweets = True
42              elif opt == '--maxtweets':
43                  tweetCriteria.maxTweets = int(arg)
44              elif opt == '--lang':
45                  tweetCriteria.lang = arg
46              elif opt == '--output':
47                  outputFileName = arg
48              elif opt == '--debug':
```

```
49                debug = True
50
51        if username_files:
52            for uf in username_files:
53                if not os.path.isfile(uf):
54                    raise Exception("File not found:%s"%uf)
55                with open(uf) as f:
56                    data = f.read()
57                    data = re.sub('(?m)#.*?$', '', data)  #
  remove comments
58                    usernames_ = [u.lstrip('@') for u in
  re.split(r'[\s,]+', data) if u]
59                    usernames_ = [u.lower() for u in
  usernames_ if u]
60                    usernames |= set(usernames_)
61                    print("Found %i usernames in %s" %
  (len(usernames_), uf))
62
63        if usernames:
64            if len(usernames) > 1:
65                tweetCriteria.username = usernames
66                if len(usernames)>20 and
  tweetCriteria.maxTweets > 0:
67                    maxtweets_ = (len(usernames) // 20 +
  (len(usernames)%20>0)) * tweetCriteria.maxTweets
68                    print("Warning: due to multiple
  username batches `maxtweets' set to %i" % maxtweets_)
69            else:
70                tweetCriteria.username = usernames.pop()
71
72        outputFile = open(outputFileName, "w+",
  encoding="utf8")
73
  outputFile.write('date,username,text,geo,mentions,hashtags,
  id\n')
74
```

```
75          cnt = 0
76          def receiveBuffer(tweets):
77              nonlocal cnt
78
79              for t in tweets:
80                  data = [t.date.strftime("%Y-%m-%d
  %H:%M:%S"),
81                      t.username,
82                      t.geo,
83                      t.mentions,
84                      t.hashtags,
85                      t.id]
86                  data[:] = [i if isinstance(i, str) else
  str(i) for i in data]
87                  outputFile.write(','.join(data) + '\n')
88
89              outputFile.flush()
90              cnt += len(tweets)
91
92              if sys.stdout.isatty():
93                  print("\rSaved %i"%cnt, end='', flush=True)
94              else:
95                  print(cnt, end=' ', flush=True)
96
97          print("Downloading tweets...")
98          got.manager.TweetManager.getTweets(tweetCriteria,
  receiveBuffer, debug=debug)
99
100     except getopt.GetoptError as err:
101         print('Arguments parser error, try -h')
102         print('\t' + str(err))
103
104     except KeyboardInterrupt:
105         print("\r\nInterrupted.\r\n")
106
107     except Exception as err:
```

```
108         print(traceback.format_exc())
109         print(str(err))
110 
111     finally:
112         if "outputFile" in locals():
113             outputFile.close()
114             print()
115             print('Done. Output file generated "%s".' %
  outputFileName)
116 
117 if __name__ == '__main__':
118     main(sys.argv[1:])
```

TweetManager.py from GetOldTweets3:

```
1  import json, re, datetime, sys, random, http.cookiejar
2  import urllib.request, urllib.parse, urllib.error
3  from pyquery import PyQuery
4  from .. import models
5 
6  class TweetManager:
7      """A class for accessing the Twitter's search engine"""
8      def __init__(self):
9          pass
10 
11     user_agents = [
12         'Mozilla/5.0 (Windows NT 6.1; WOW64; rv:63.0)
   Gecko/20100101 Firefox/63.0',
13         'Mozilla/5.0 (Windows NT 6.1; WOW64; rv:62.0)
   Gecko/20100101 Firefox/62.0',
14         'Mozilla/5.0 (Windows NT 6.1; WOW64; rv:61.0)
   Gecko/20100101 Firefox/61.0',
15         'Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:63.0)
   Gecko/20100101 Firefox/63.0',
16         'Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36
   (KHTML, like Gecko) Chrome/70.0.3538.77 Safari/537.36',
17         'Mozilla/5.0 (Windows NT 6.3; Win64; x64)
```

```
    AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.77
    Safari/537.36',
18          'Mozilla/5.0 (Windows NT 6.1; Trident/7.0; rv:11.0)
    like Gecko',
19          'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6)
    AppleWebKit/605.1.15 (KHTML, like Gecko) Version/12.0
    Safari/605.1.15',
20      ]
21
22      @staticmethod
23      def getTweets(tweetCriteria, receiveBuffer=None,
    bufferLength=100, proxy=None, debug=False):
24          """Get tweets that match the tweetCriteria
    parameter
25          A static method.
26
27          Parameters
28          ----------
29          tweetCriteria : tweetCriteria, an object that
    specifies a match criteria
30          receiveBuffer : callable, a function that will be
    called upon a getting next `bufferLength' tweets
31          bufferLength: int, the number of tweets to pass to
    `receiveBuffer' function
32          proxy: str, a proxy server to use
33          debug: bool, output debug information
34          """
35          results = []
36          resultsAux = []
37          cookieJar = http.cookiejar.CookieJar()
38          user_agent =
    random.choice(TweetManager.user_agents)
39
40          all_usernames = []
41          usernames_per_batch = 20
42
43          if hasattr(tweetCriteria, 'username'):
```

```
44              if type(tweetCriteria.username) == str or not
    hasattr(tweetCriteria.username, '__iter__'):
45                  tweetCriteria.username =
    [tweetCriteria.username]
46
47              usernames_ = [u.lstrip('@') for u in
    tweetCriteria.username if u]
48              all_usernames = sorted({u.lower() for u in
    usernames_ if u})
49              n_usernames = len(all_usernames)
50              n_batches = n_usernames // usernames_per_batch
    + (n_usernames % usernames_per_batch > 0)
51          else:
52              n_batches = 1
53
54          for batch in range(n_batches):  # process
    all_usernames by batches
55              refreshCursor = ''
56              batch_cnt_results = 0
57
58              if all_usernames:  # a username in the
    criteria?
59                  tweetCriteria.username =
    all_usernames[batch*usernames_per_batch:batch*usernames_per
    _batch+usernames_per_batch]
60
61              active = True
62              while active:
63                  json =
    TweetManager.getJsonResponse(tweetCriteria, refreshCursor,
    cookieJar, proxy, user_agent, debug=debug)
64                  if len(json['items_html'].strip()) == 0:
65                      break
66
67                  refreshCursor = json['min_position']
68                  scrapedTweets = PyQuery(json['items_html'])
69                  #Remove incomplete tweets withheld by
```

```
    Twitter Guidelines
70                  scrapedTweets.remove('div.withheld-tweet')
71                  tweets =
    scrapedTweets('div.js-stream-tweet')
72
73                  if len(tweets) == 0:
74                      break
75
76                  for tweetHTML in tweets:
77                      tweetPQ = PyQuery(tweetHTML)
78                      tweet = models.Tweet()
79
80                      usernames =
    tweetPQ("span.username.u-dir b").text().split()
81                      if not len(usernames):  # fix for issue
    #13
82                          continue
83
84                      tweet.username = usernames[0]
85                      tweet.to = usernames[1] if
    len(usernames) >= 2 else None  # take the first recipient if
    many
86                      tweet.text = re.sub(r"\s+", " ",
    tweetPQ("p.js-tweet-text").text())\
87                          .replace('# ', '#').replace('@ ',
    '@').replace('$ ', '$')
88                      tweet.retweets =
    int(tweetPQ("span.ProfileTweet-action--retweet
    span.ProfileTweet-actionCount").attr("data-tweet-stat-count"
    ).replace(",", ""))
89                      tweet.favorites =
    int(tweetPQ("span.ProfileTweet-action--favorite
    span.ProfileTweet-actionCount").attr("data-tweet-stat-count"
    ).replace(",", ""))
90                      tweet.replies =
    int(tweetPQ("span.ProfileTweet-action--reply
    span.ProfileTweet-actionCount").attr("data-tweet-stat-count"
```

```
).replace(",", ""))
                    tweet.id =
tweetPQ.attr("data-tweet-id")
                    tweet.permalink = 'https://twitter.com'
+ tweetPQ.attr("data-permalink-path")
                    tweet.author_id =
int(tweetPQ("a.js-user-profile-link").attr("data-user-id"))

                    dateSec = int(tweetPQ("small.time
span.js-short-timestamp").attr("data-time"))
                    tweet.date =
datetime.datetime.fromtimestamp(dateSec,
tz=datetime.timezone.utc)
                    tweet.formatted_date =
datetime.datetime.fromtimestamp(dateSec,
tz=datetime.timezone.utc)\

.strftime("%a %b %d %X +0000 %Y")
                    tweet.mentions = "
".join(re.compile('(@\\w*)').findall(tweet.text))
                     tweet.hashtags = "
".join(re.compile('(#\\w*)').findall(tweet.text))

                    geoSpan = tweetPQ('span.Tweet-geo')
                    if len(geoSpan) > 0:
                        tweet.geo = geoSpan.attr('title')
                    else:
                        tweet.geo = ''

                    urls = []
                    for link in tweetPQ("a"):
                        try:

urls.append((link.attrib["data-expanded-url"]))
                        except KeyError:
                            pass

```

```
115                     tweet.urls = ",".join(urls)
116
117                     results.append(tweet)
118                     resultsAux.append(tweet)
119
120                     if receiveBuffer and len(resultsAux) >=
  bufferLength:
121                         receiveBuffer(resultsAux)
122                         resultsAux = []
123
124                     batch_cnt_results += 1
125                     if tweetCriteria.maxTweets > 0 and
  batch_cnt_results >= tweetCriteria.maxTweets:
126                         active = False
127                         break
128
129             if receiveBuffer and len(resultsAux) > 0:
130                 receiveBuffer(resultsAux)
131                 resultsAux = []
132
133         return results
134
135     @staticmethod
136     def getJsonResponse(tweetCriteria, refreshCursor,
  cookieJar, proxy, useragent=None, debug=False):
137         """Invoke an HTTP query to Twitter.
138         Should not be used as an API function. A static
  method.
139         """
140         url = "https://twitter.com/i/search/timeline?"
141
142         if not tweetCriteria.topTweets:
143             url += "f=tweets&"
144
145         url += ("vertical=news&q=%s&src=typd&%s"
146
  "&include_available_features=1&include_entities=1&max_posit
```

```
  ion=%s"
147                 "&reset_error_state=false")
148
149         urlGetData = ''
150
151         if hasattr(tweetCriteria, 'querySearch'):
152             urlGetData += tweetCriteria.querySearch
153
154         if hasattr(tweetCriteria, 'username'):
155             if not hasattr(tweetCriteria.username,
  '__iter__'):
156                 tweetCriteria.username =
  [tweetCriteria.username]
157
158             usernames_ = [u.lstrip('@') for u in
  tweetCriteria.username if u]
159             tweetCriteria.username = {u.lower() for u in
  usernames_ if u}
160
161             usernames = [' from:'+u for u in
  sorted(tweetCriteria.username)]
162             if usernames:
163                 urlGetData += ' OR'.join(usernames)
164
165         if hasattr(tweetCriteria, 'since'):
166             urlGetData += ' since:' + tweetCriteria.since
167
168         if hasattr(tweetCriteria, 'until'):
169             urlGetData += ' until:' + tweetCriteria.until
170
171         if hasattr(tweetCriteria, 'lang'):
172             urlLang = 'l=' + tweetCriteria.lang + '&'
173         else:
174             urlLang = ''
175         url = url %
  (urllib.parse.quote(urlGetData.strip()), urlLang,
  urllib.parse.quote(refreshCursor))
```

```
176         useragent = useragent or
  TweetManager.user_agents[0]
177 
178         headers = [
179             ('Host', "twitter.com"),
180             ('User-Agent', useragent),
181             ('Accept', "application/json, text/javascript,
  */*; q=0.01"),
182             ('Accept-Language', "en-US,en;q=0.5"),
183             ('X-Requested-With', "XMLHttpRequest"),
184             ('Referer', url),
185             ('Connection', "keep-alive")
186         ]
187 
188         if proxy:
189             opener =
  urllib.request.build_opener(urllib.request.ProxyHandler({'h
  ttp': proxy, 'https': proxy}),
  urllib.request.HTTPCookieProcessor(cookieJar))
190         else:
191             opener =
  urllib.request.build_opener(urllib.request.HTTPCookieProces
  sor(cookieJar))
192         opener.addheaders = headers
193 
194         if debug:
195             print(url)
196             print('\n'.join(h[0]+': '+h[1] for h in
  headers))
197 
198         try:
199             response = opener.open(url)
200             jsonResponse = response.read()
201         except Exception as e:
202             print("An error occured during an HTTP
  request:", str(e))
203             print("Try to open in browser:
```

```
    https://twitter.com/search?q=%s&src=typd" %
    urllib.parse.quote(urlGetData))
204             sys.exit()
205
206         try:
207             s_json = jsonResponse.decode()
208         except:
209             print("Invalid response from Twitter")
210             sys.exit()
211
212         try:
213             dataJson = json.loads(s_json)
214         except:
215             print("Error parsing JSON: %s" % s_json)
216             sys.exit()
217
218         if debug:
219             print(s_json)
220             print("---\n")
221
222         return dataJson
```

TweetCriteria.py from GetOldTweets3

```
1  class TweetCriteria:
2      """Search parameters class"""
3
4      def __init__(self):
5          self.maxTweets = 0
6          self.topTweets = False
7          self.within = "15mi"
8
9      def setUsername(self, username):
10         """Set username(s) of tweets author(s)
11         Parameters
12         ----------
13         username : str or iterable
```

```
14
15          If `username' is specified by str it should be a
    single username or
16          usernames separeated by spaces or commas.
17          `username` can contain a leading @
18
19          Examples:
20              setUsername('barackobama')
21              setUsername('barackobama,whitehouse')
22              setUsername('barackobama whitehouse')
23              setUsername(['barackobama','whitehouse'])
24          """
25          self.username = username
26          return self
27
28      def setSince(self, since):
29          """Set a lower bound date in UTC
30          Parameters
31          ----------
32          since : str,
33                  format: "yyyy-mm-dd"
34          """
35          self.since = since
36          return self
37
38      def setUntil(self, until):
39          """Set an upper bound date in UTC (not included in
    results)
40          Parameters
41          ----------
42          until : str,
43                  format: "yyyy-mm-dd"
44          """
45          self.until = until
46          return self
47
```

```
48      def setQuerySearch(self, querySearch):
49          """Set a text to be searched for
50          Parameters
51          ----------
52          querySearch : str
53          """
54          self.querySearch = querySearch
55          return self
56
57      def setMaxTweets(self, maxTweets):
58          """Set the maximum number of tweets to search
59          Parameters
60          ----------
61          maxTweets : int
62          """
63          self.maxTweets = maxTweets
64          return self
65
66      def setLang(self, Lang):
67          """Set language
68          Parameters
69          ----------
70          Lang : str
71          """
72          self.lang = Lang
73          return self
74
75      def setTopTweets(self, topTweets):
76          """Set the flag to search only for top tweets
77          Parameters
78          ----------
79          topTweets : bool
80          """
81          self.topTweets = topTweets
82          return self
```

Data cleaning and Extraction of Hashtags

Clean.py

```
import preprocessor as p
import numpy as np
import pandas as pd
import re as re
import types
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove those credentials before you share the notebook.
client_59ce70433d0c4b4d95817a77b9042741 = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='OHfo7peHW5YcrP1k2sp-BWA8c5HUs53HUEiUS4sqYYmT',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.eu-geo.objectstorage.service.networklayer.com')


body = client_59ce70433d0c4b4d95817a77b9042741.get_object(Bucket='megha1-donotdelete-pr-k9tzkndbrlckzs',Key='data_2020-03-01.csv')['Body']
# add missing __iter__ method, so pandas accepts body as
```

```
    file-like object
24 if not hasattr(body, "__iter__"): body.__iter__ =
   types.MethodType( __iter__, body )
25
26 def preprocess_tweet(row):
27     text = row['Tweet Content']
28     text = p.clean(text)
29     return text
30
31 parse_dt = pd.read_csv(body)
32 parse_dt=parse_dt[1:]
33
34 p.set_options(p.OPT.URL, p.OPT.MENTION)
35 parse_dt['Tweet Content']=parse_dt.apply(preprocess_tweet,
   axis=1)
36 print(parse_dt)
37 ls=[]
38 for row in parse_dt['Tweet Content']:
39     for ext in re.findall(r"#(\w+)", row):
40         ls.append(ext)
41 print(ls)
```

Analysis using NLTK and Naive Bayes

```
1  from nltk.stem.wordnet import WordNetLemmatizer
2  from nltk.corpus import twitter_samples, stopwords
3  from nltk.tag import pos_tag
4  from nltk.tokenize import word_tokenize
5  from nltk import FreqDist, classify, NaiveBayesClassifier
6  nltk.download('punkt')
7
8  import re, string, random
9
10
11
12 def remove_noise(tweet_tokens, stop_words = ()):
13
```

```
14      cleaned_tokens = []
15
16      for token, tag in pos_tag(tweet_tokens):
17          token =
    re.sub('http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+#]|[!*\(\),]|'
    \
18                         '(?:%[0-9a-fA-F][0-9a-fA-F]))+','',
    token)
19          token = re.sub("(@[A-Za-z0-9_]+)","", token)
20
21          if tag.startswith("NN"):
22              pos = 'n'
23          elif tag.startswith('VB'):
24              pos = 'v'
25          else:
26              pos = 'a'
27
28          lemmatizer = WordNetLemmatizer()
29          token = lemmatizer.lemmatize(token, pos)
30
31          if len(token) > 0 and token not in
    string.punctuation and token.lower() not in stop_words:
32              cleaned_tokens.append(token.lower())
33      return cleaned_tokens
34
35  def get_all_words(cleaned_tokens_list):
36      for tokens in cleaned_tokens_list:
37          for token in tokens:
38              yield token
39
40  def get_tweets_for_model(cleaned_tokens_list):
41      for tweet_tokens in cleaned_tokens_list:
42          yield dict([token, True] for token in tweet_tokens)
43
44  if __name__ == "__main__":
45
46      positive_tweets =
```

```
  twitter_samples.strings('positive_tweets.json')
47    negative_tweets =
  twitter_samples.strings('negative_tweets.json')
48    text =
  twitter_samples.strings('tweets.20150430-223406.json')
49    #tweet_tokens =
  twitter_samples.tokenized('positive_tweets.json')[0]
50
51    stop_words = stopwords.words('english')
52
53    positive_tweet_tokens =
  twitter_samples.tokenized('positive_tweets.json')
54    negative_tweet_tokens =
  twitter_samples.tokenized('negative_tweets.json')
55
56    positive_cleaned_tokens_list = []
57    negative_cleaned_tokens_list = []
58
59    for tokens in positive_tweet_tokens:
60
  positive_cleaned_tokens_list.append(remove_noise(tokens,
  stop_words))
61
62    for tokens in negative_tweet_tokens:
63
  negative_cleaned_tokens_list.append(remove_noise(tokens,
  stop_words))
64
65    all_pos_words =
  get_all_words(positive_cleaned_tokens_list)
66
67    freq_dist_pos = FreqDist(all_pos_words)
68    print(freq_dist_pos.most_common(10))
69
70    positive_tokens_for_model =
  get_tweets_for_model(positive_cleaned_tokens_list)
71    negative_tokens_for_model =
```

```
    get_tweets_for_model(negative_cleaned_tokens_list)
72
73      positive_dataset = [(tweet_dict, "Positive")
74                          for tweet_dict in
    positive_tokens_for_model]
75
76      negative_dataset = [(tweet_dict, "Negative")
77                          for tweet_dict in
    negative_tokens_for_model]
78
79      dataset = positive_dataset + negative_dataset
80
81      random.shuffle(dataset)
82
83      train_data = dataset[:7000]
84      test_data = dataset[7000:]
85
86      classifier = NaiveBayesClassifier.train(train_data)
87
88      print("Accuracy is:", classify.accuracy(classifier,
    test_data))
89
90      print(classifier.show_most_informative_features(10))
91
92      custom_tweet = "I ordered just once from TerribleCo,
    they screwed up, never used the app again."
93
94      custom_tokens =
    remove_noise(word_tokenize(custom_tweet))
95
96      print(custom_tweet, classifier.classify(dict([token,
    True] for token in custom_tokens)))
```

upload.py

```
1  import csv
2  csvfile = open('new.csv', 'w')
```

```
csvwriter = csv.writer(csvfile)
for item in l:
    csvwriter.writerow(item)
csvfile.close()


# The following code contains the credentials for a file in your IBM Cloud Object Storage.
# You might want to remove those credentials before you share your notebook.
# @hidden_cell
# The following code contains the credentials for a file in your IBM Cloud Object Storage.
# You might want to remove those credentials before you share your notebook.
# @hidden_cell
# The following code contains the credentials for a file in your IBM Cloud Object Storage.
# You might want to remove those credentials before you share your notebook.
credentials = {
    'IAM_SERVICE_ID': 'iam-ServiceId-1d64fb0c-cd35-4a4b-8183-4245ebe3cc6e',
    'IBM_API_KEY_ID': 'OHfo7peHW5YcrP1k2sp-BWA8c5HUs53HUEiUS4sqYYmT',
    'ENDPOINT': 'https://s3.eu-geo.objectstorage.service.networklayer.com',
    'IBM_AUTH_ENDPOINT': 'https://iam.cloud.ibm.com/oidc/token',
    'BUCKET': 'megha1-donotdelete-pr-k9tzkndbrlckzs',
    'FILE': 'e3.csv'
}
cos = ibm_boto3.client(service_name='s3',
    ibm_api_key_id=credentials['IBM_API_KEY_ID'],
    ibm_service_instance_id=credentials['IAM_SERVICE_ID'],
    ibm_auth_endpoint=credentials['IBM_AUTH_ENDPOINT'],
```

```
    config=Config(signature_version='oauth'),
    endpoint_url=credentials['ENDPOINT'])
cos.upload_file(Filename='new.csv',Bucket=credentials['BUCKET'],Key='e3.csv')
print('done upload')
```

Analysis using VADER

```
import ibm_boto3
from botocore.client import Config
import pandas as pd


# @hidden_cell
# The following code contains the credentials for a file in your IBM Cloud Object Storage.
# You might want to remove those credentials before you share your notebook.
credentials_1 = {
    'IAM_SERVICE_ID': 'iam-ServiceId-0ec16a92-04f6-49d2-b6b4-ed28d9517059',
    'IBM_API_KEY_ID': '6LEOLcwLZH2OWchPeVQDITDWicnDMhUASHOt_jDikuB2',
    'ENDPOINT': 'https://s3.eu-geo.objectstorage.service.networklayer.com',
    'IBM_AUTH_ENDPOINT': 'https://iam.cloud.ibm.com/oidc/token',
    'BUCKET': 'vader-donotdelete-pr-cyihqbuizle5xb',
    'FILE': 'NRC-Emotion-Intensity-Lexicon-v1.txt'
}

cos = ibm_boto3.client('s3',

ibm_api_key_id=credentials_1['IBM_API_KEY_ID'],

ibm_service_instance_id=credentials_1['IAM_SERVICE_ID'],

```

```
    ibm_auth_endpoint=credentials_1['IBM_AUTH_ENDPOINT'],
22
    config=Config(signature_version='oauth'),
23
    endpoint_url=credentials_1['ENDPOINT'])
24
25  def get_file(filename):
26      '''Retrieve file from Cloud Object Storage'''
27      fileobject =
    cos.get_object(Bucket=credentials_1['BUCKET'],
    Key=filename)['Body']
28      return fileobject
29
30  fp = get_file('NRC-Emotion-Intensity-Lexicon-v1.txt')
31  emolex_df = pd.read_csv(fp,names=["word", "emotion",
    "association"],sep='\t')
32  emolex_df = emolex_df[1:]
33  print(emolex_df.head())
34
35  emolex_words = emolex_df.pivot(index='word',
36                                 columns='emotion',
37
    values='association').reset_index()
38  print(emolex_words.head())
39  emotions = emolex_words.columns.drop('word')
40  print(emotions)
41  import nltk
42  from nltk.corpus import stopwords
43  nltk.download('stopwords')
44  nltk.download('punkt')
45  nltk.download('wordnet')
46  nltk.download('averaged_perceptron_tagger')
47  from nltk.tag import pos_tag
48  from nltk import word_tokenize
49  from nltk.stem.wordnet import WordNetLemmatizer
50  import re, string
51  lemmatizer = WordNetLemmatizer()
```

```
52
53 def remove_noise(tweet_tokens, stop_words = ()):
54
55     cleaned_tokens = []
56
57     for token, tag in pos_tag(tweet_tokens):
58         #token =
   re.sub('http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+#]|[!*\(\),]|'
   \
59         #                '(?:%[0-9a-fA-F][0-9a-fA-F]))+','',
   token)
60         #token = re.sub('(@[A-Za-z0-9_]+)','', token)
61
62         if tag.startswith("NN"):
63             pos = 'n'
64         elif tag.startswith('VB'):
65             pos = 'v'
66         else:
67             pos = 'a'
68
69
70         token = lemmatizer.lemmatize(token, pos)
71
72         if len(token) > 0 and token not in
   string.punctuation and token.lower() not in
   stopwords.words():
73             cleaned_tokens.append(token.lower())
74     return cleaned_tokens
75 import types
76 import pandas as pd
77 from botocore.client import Config
78 import ibm_boto3
79
80 def __iter__(self): return 0
81
82 # @hidden_cell
83 # The following code accesses a file in your IBM Cloud
```

```
  Object Storage. It includes your credentials.
84 # You might want to remove those credentials before you
  share the notebook.
85 client_59ce70433d0c4b4d95817a77b9042741 =
  ibm_boto3.client(service_name='s3',
86
  ibm_api_key_id='OHfo7peHW5YcrP1k2sp-BWA8c5HUs53HUEiUS4sqYYm
  T',
87
  ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
88     config=Config(signature_version='oauth'),
89
  endpoint_url='https://s3.eu-geo.objectstorage.service.netwo
  rklayer.com')
90
91 body =
  client_59ce70433d0c4b4d95817a77b9042741.get_object(Bucket='
  megha1-donotdelete-pr-k9tzkndbrlckzs',Key='combined_june_fin
  al.csv')['Body']
92 # add missing __iter__ method, so pandas accepts body as
  file-like object
93 if not hasattr(body, "__iter__"): body.__iter__ =
  types.MethodType( __iter__, body )
94
95 jdf = pd.read_csv(body)
96 jdf.head()
97 import preprocessor as p
98 import numpy as np
99 nltk.download('stopwords')
100 stop_words = stopwords.words('english')
101
102 #p.set_options(p.OPT.URL,p.OPT.MENTION)
103 p.set_options(p.OPT.URL,p.OPT.MENTION,p.OPT.NUMBER,p.OPT.H
  ASHTAG, p.OPT.EMOJI)
104 emo_df = pd.DataFrame(0, index=jdf.index,
  columns=emotions)
105 new_jdf = pd.DataFrame()
```

```
new_jdf['Tweet'] = jdf['Tweet Content']
for emotion in emotions:
    new_jdf[emotion] = 0.0
#print(new_jdf.head())
def sentiment_score(row):
    tweet = row['Tweet']
    tweet = p.clean(tweet)
    tokens = remove_noise(set(word_tokenize(tweet)),stop_words)

    for token in tokens:
        #print(token)
        emo_score = emolex_words[emolex_words.word == token]
        if not emo_score.empty:
            for emotion in list(emotions):
                if not emo_score[emotion].isna().bool():
                    row[emotion] = float(row[emotion]) + float(emo_score[emotion].iloc[0])
    return row
print('hi')
```

```
#Testing the code for 100 tweet samples
l = []
#print(sentiment_score(new_jdf.iloc[27]))
for i in range(0,101):
    #print(sentiment_score(new_jdf.iloc[i]))
    l.append(sentiment_score(new_jdf.iloc[i]))
#new_jdf = new_jdf.apply(sentiment_score,axis=1)
print('done')
print(l[:5])
```

Visualization code

```
import plotly.graph_objs as go
import numpy as np
import matplotlib.pyplot as plt
import types
import pandas as pd
from botocore.client import Config
import ibm_boto3
import types
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove those credentials before you share the notebook.
client_59ce70433d0c4b4d95817a77b9042741 = ibm_boto3.client(service_name='s3',

ibm_api_key_id='OHfo7peHW5YcrP1k2sp-BWA8c5HUs53HUEiUS4sqYYmT',

ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),

endpoint_url='https://s3.eu-geo.objectstorage.service.networklayer.com')

body = client_59ce70433d0c4b4d95817a77b9042741.get_object(Bucket='
```

```
    megha1-donotdelete-pr-k9tzkndbrlckzs',Key='e2.csv')['Body']
25  # add missing __iter__ method, so pandas accepts body as
    file-like object
26  if not hasattr(body, "__iter__"): body.__iter__ =
    types.MethodType( __iter__, body )
27
28  june = pd.read_csv(body)
29  print('hi')
30  subset_june = june[june["Label"]=="Positive"]
31  Pcolumn_count = subset_june['Label'].count()
32  print(Pcolumn_count)
33  print('pos done')
34  subset_june = june[june["Label"]=="Negative"]
35  Ncolumn_count = subset_june['Label'].count()
36  print(Ncolumn_count)
37  print('neg done')
38
39  #import matplotlib.pyplot as plt
40  label_x = ['June_pos', 'June_neg']
41  count = [Pcolumn_count, Ncolumn_count]
42
43  x_pos = [i for i, _ in enumerate(label_x)]
44
45  plt.bar(x_pos, count, color='green')
46  plt.xlabel("Month")
47  plt.ylabel("No of tweets")
48  plt.title("Sentiment analysis")
49  plt.plot()
```