Kevin Lin
CISC 4900

Project Summary

The topic of this project is about data analysis using the social media platform twitter. Using Twitter's API and token keys, tweets were extracted from twitter. A python library called tweepy was used in this process to extract the tweets. The Listener class that was created to inherit the Tweepy's Stream Listener class. This allowed the live extraction of tweets. A streamer class was created to verify the API and token keys. A constructor class was created for the time limit and an array was created for the tweets in JSON style. The on_data method was created to fetch for live tweets. The on_status and on_error methods were created to check if there were live tweets fetched. Two constructors were created for the Streamer class One constructor was used for the live tweets, the other was used to get previous tweets. A filter method was created to filter out any unwanted previous tweets. All these methods and classes are in the Twitter_Stream.py file.

Twitter_Stream.py

```python
import time
import csv
from tweepy import OAuthHandler
from tweepy import Stream
from tweepy import API
from tweepy import Cursor
import Twitter_Credentials
from tweepy.streaming import StreamListener
import pandas as pd


# This class handles whether or not the authentication works.
class Listener(StreamListener):

    # constructor sets up the current time and the end time of the stream
    def __init__(self, time_limit=60):
        self.start_time = time.time()  # instantiates the current time
        self.end_time = time_limit  # instantiates the end time in seconds
        self.twitter_data = []  # instantiates a new list for JSON data

    def on_data(self, raw_data):
        try:
            if (time.time() - self.start_time) < self.end_time:  # calculates the amount of seconds left in the stream
                self.twitter_data.append(raw_data)  # writes JSON data into list
                return True
        except BaseException as e:
            print('failed because,', str(e))
            time.sleep(5)
            pass
        json_file = open('unfiltered_tweets_Donald_Trump.json', 'a', encoding='utf-8',
                         newline='')  # writes into JSON file
        json_file.write(u'[\n')
        json_file.write(',\'.join(self.twitter_data))  # writes into JSON file with the list
        json_file.write(u'\n]')
```

Twitter_Stream.py

```python
        json_file.write(u'\n]')
        json_file.close()
        exit()

    # If the connection works it will return true
    def on_status(self, status):
        print(status)
        return True

    # If there is a connection problem it will return false
    def on_error(self, status_code):
        if status_code == 420:
            # Will return false if too many login attempts. Too many times authenticating with the account name.
            return False
        print(status_code)


# This class authenticates twitter's API key and access token. It also output tweets.
class Streamer:

    # The constructor authenticates the API key and access token. This constructor start the time limit for live tweets
    def __init__(self, time_limit):
        self.listen = Listener(time_limit=time_limit)
        self.auth = OAuthHandler(Twitter_Credentials.API_KEY, Twitter_Credentials.API_KEY_SECRET)
        self.auth.set_access_token(Twitter_Credentials.ACCESS_TOKEN, Twitter_Credentials.ACCESS_TOKEN_SECRET)

    # This constructor authenticates the API key and access token. This constructor searches for previous tweets
    def __init__(self):
        self.auth = OAuthHandler(Twitter_Credentials.API_KEY, Twitter_Credentials.API_KEY_SECRET)
        self.auth.set_access_token(Twitter_Credentials.ACCESS_TOKEN, Twitter_Credentials.ACCESS_TOKEN_SECRET)

    # This method output any tweets that are related to the user's input
    def output_live_tweets(self, input_filter):
        streamer = Stream(self.auth, self.listen)
```

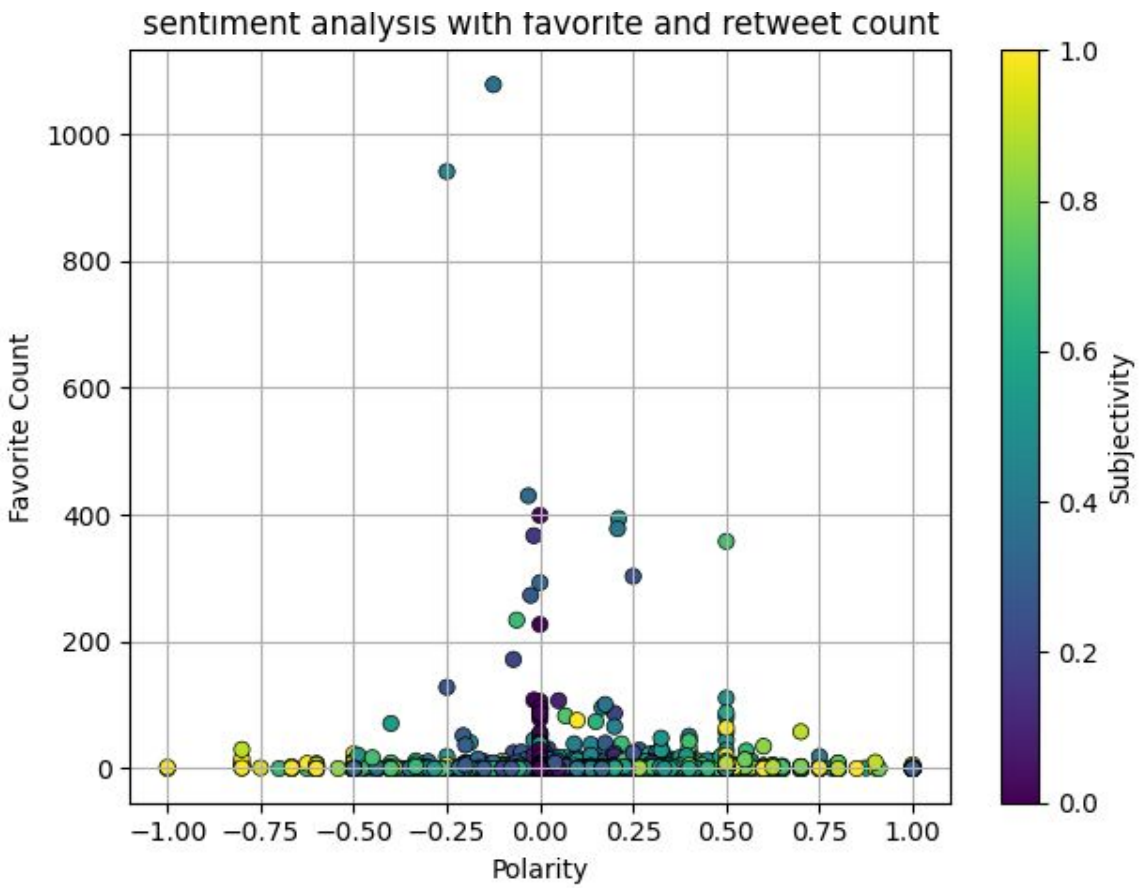
```python
    # This method output any tweets that are related to the user's input
    def output_live_tweets(self, input_filter):
        streamer = Stream(self.auth, self.listen)
        streamer.filter(track=input_filter)

    def output_previous_tweets(self,start, end):
        api = API(self.auth, wait_on_rate_limit=True)
        csv_file = open('Voting.csv', 'w', newline='', encoding="utf-8") #tweets_data_privacy
        writer = csv.writer(csv_file)  # opens csv file
        fieldnames = ['created at', 'text', 'name',
                      'screen name', 'location',
                      'verified', 'followers count', 'friends count',
                      'listed count', 'total favorites count', 'statuses count',
                      'account creation date', 'retweet count',
                      'favorite count']  # fieldnames
        writer.writerow(fieldnames)  # write fieldnames into header of the csv file
        # fetches for tweets based on keywords and filters out any retweets. Tweepy's API will filter out user
        # information.
        for tweet in Cursor(api.search, q="Voting Election -filter:retweets AND -filter:replies", lang="en", since=start,
                            until=end).items(150000):
            writer.writerow([tweet.created_at, tweet.text.encode('unicode_escape'),
                             tweet.user.name, tweet.user.screen_name, tweet.user.location,
                             tweet.user.verified, tweet.user.followers_count,
                             tweet.user.friends_count, tweet.user.listed_count,
                             tweet.user.favourites_count, tweet.user.statuses_count,
                             tweet.user.created_at,tweet.retweet_count, tweet.favorite_count])
```

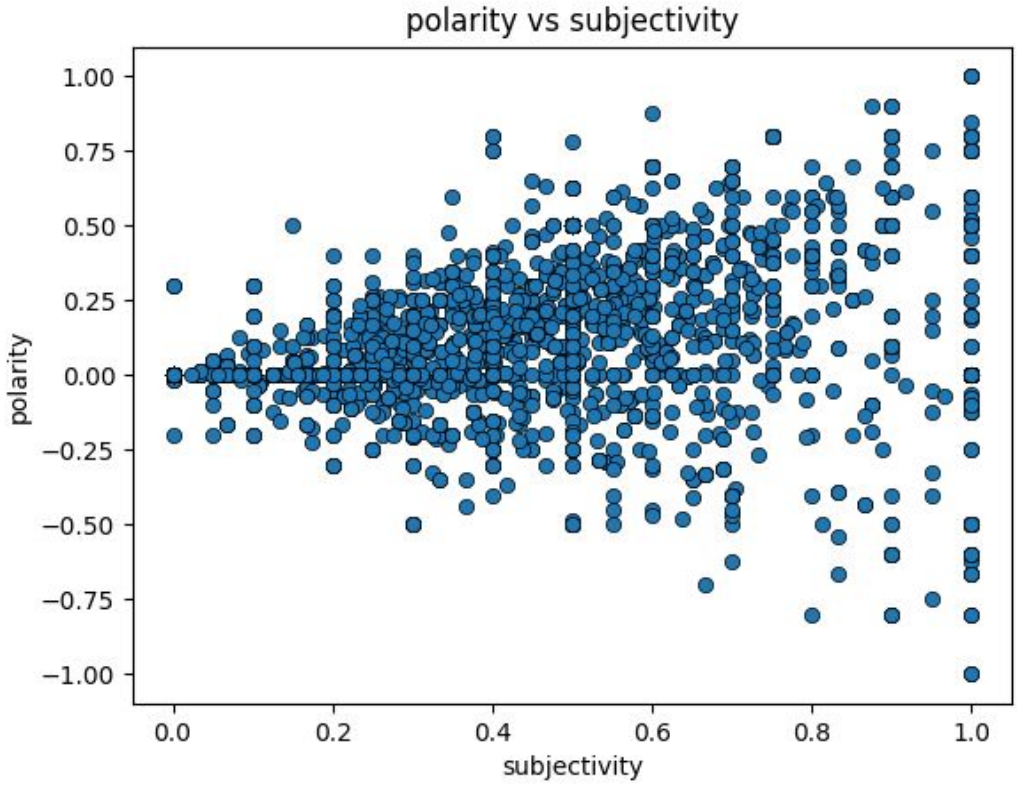
```python
    # this method filters out anything unrelated.
    def filter(self):
        read = pd.read_csv("Voting.csv")  # reads in CSV file tweets_data_privacy
        df = pd.DataFrame(read)  # turns CSV file into data frame
        new_data = df[~df['name'].str.contains('news|News|.com|EducationLaw|Honey|Roger Verhoeven|'
                                               'Policy Pro\'s - Business Policy Writers|Coca-Cola GB|JD Supra|'
                                               'Privacy Watch|Giridhar|QV Develop|Data Vault|Privacy Professionals Briefly|'
                                               'PrivacyPro b|Privacy Tech Briefly|PrivacyTech b|Hacking Essentials|Lost Coast Outpost|'
                                               'NortonRoseFulbright|Urdhva Tech|SGV Tribune|O.C. Register|Daniel Haran|Anas Najeeb|report|'
                                               'Report|SAP Customer Data Cloud|ResponSight')]
        # filters any names that contain any words that match with the words above.
        new_data.to_csv('2.csv', index=False)  # returns a new CSV file filtered_tweets
```

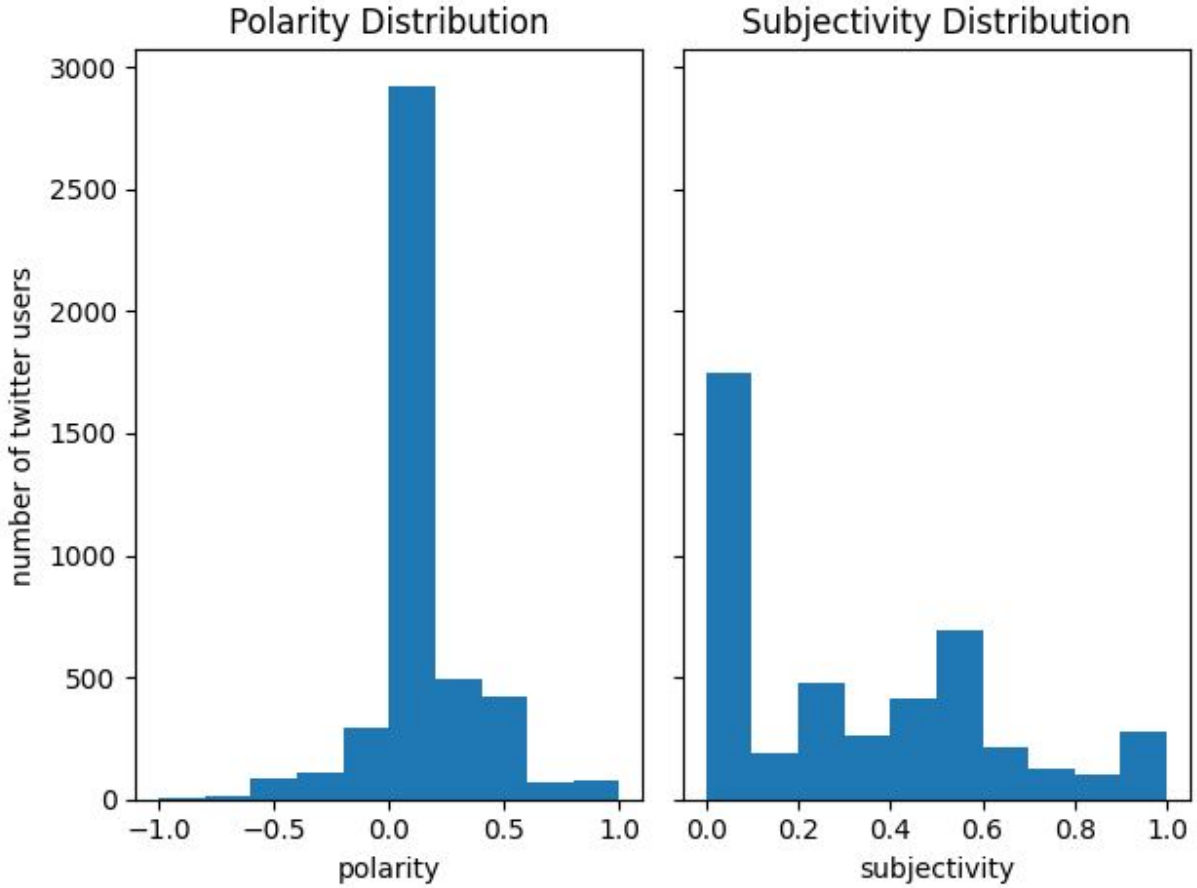
A JSON_TO_CSV.py and organize method was created to convert the live tweets from JSON to CSV files. The reason for making JSON_TO_CSV into another file is, during run time if you run output_live_tweets and organize methods in the same file, it will give a value error.

```python
import json
import csv


def organize():
    json_file = open('unfiltered_tweets_Donald_Trump.json', 'r', encoding="utf-8").read()  # reads the JSON file
    json_data = json.loads(json_file)  # string becomes a json Python object
    csv_file = open('filtered_tweets_Donald_Trump.csv', 'w', encoding="utf-8", newline='')  # opens csv file
    writer = csv.writer(csv_file)  # creates csv writer object
    fieldnames = ['created at', 'text', 'screen name',
                  'location', 'verified', 'followers',
                  'friends']  # fieldnames
    writer.writerow(fieldnames)  # writes fieldnames on the first row

    # gets fields from the json object and writes a row into csv file
    for tweet in json_data:
        if tweet.get('lang') != "en":  # if the tweet is not in english, don't write in csv file
            continue
        else:
            writer.writerow([tweet.get('created_at'), tweet.get('text').encode('unicode_escape'),
                             tweet.get('user').get('screen_name'), tweet.get('user').get('location'),
                             tweet.get('user').get('verified'), tweet.get('user').get('followers_count'),
                             tweet.get('user').get('friends_count')])
    csv_file.close()


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

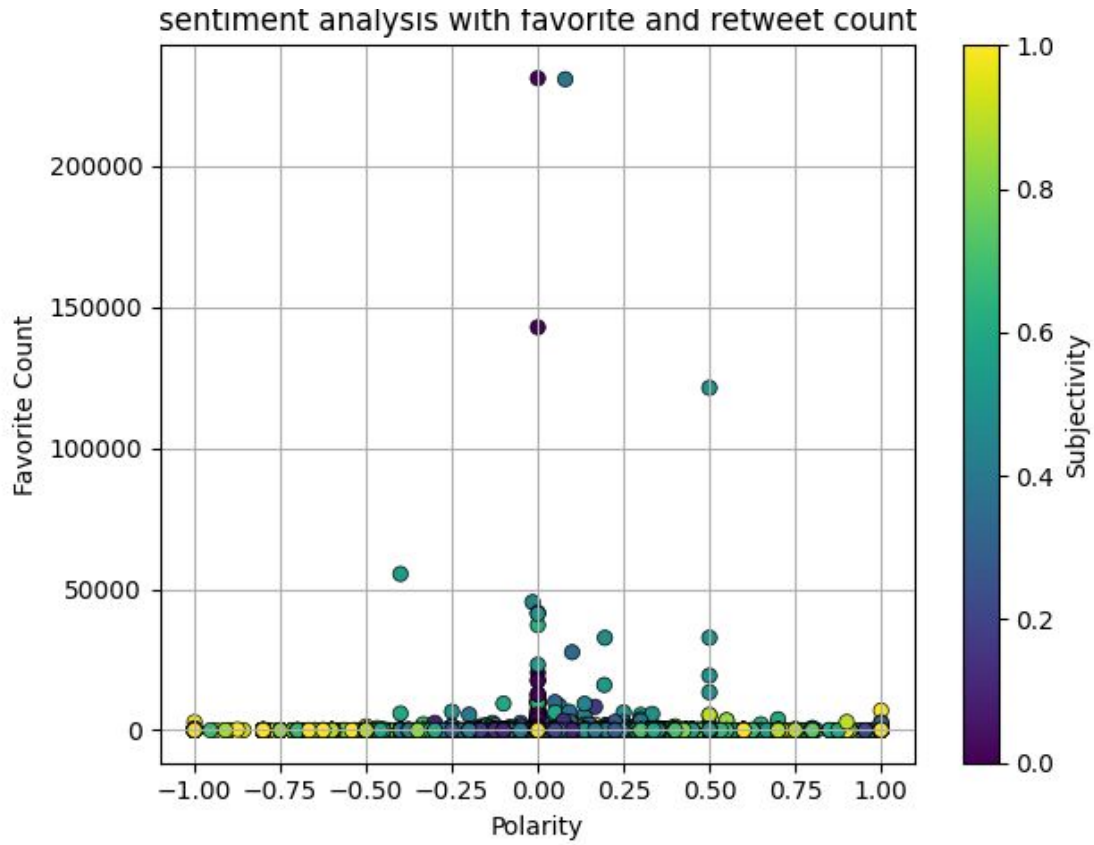
Next, we created a Sentiment_analysis.py file with a class called Analysis. The first method is called polarity_and_subjectivity. The tweets that were extracted were used to detect the polarity and subjectivity of the text. Then, polarity and subjectivity are classified into three ranges and will create a new csv file. The next three methods help create a scatter or histogram plot. The subjectivity_class method allows a subjectivity class column to classify the subjectivity into three classes(0,1,2) based on the three ranges. The to_classification methods take the features, followers count, friends count, retweet count, favorite count, polarity, SubjectivityClass and make a new csv file. The new csv file will be used for a machine learning classification algorithm to determine whether we can predict subjectivity. Based on the data extract from twitter, there are six charts represented for the data. Three are represented for the topic, data privacy and the other three for the 2020 election voting fiasco.
Data privacy:

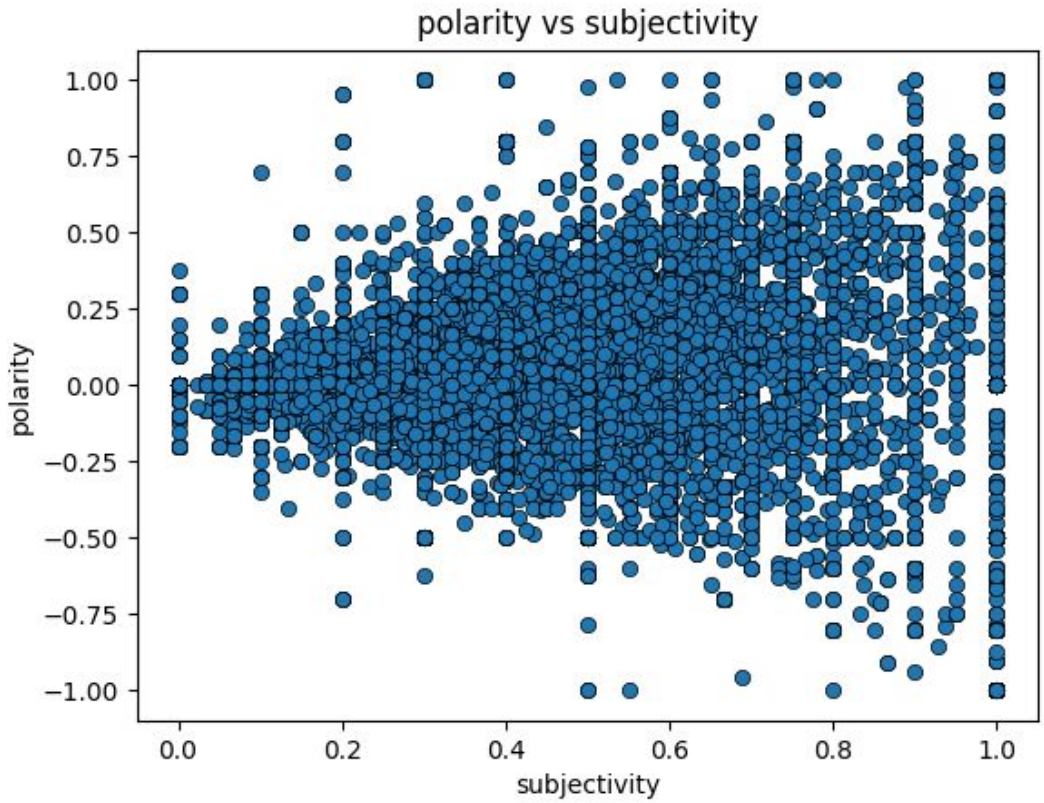


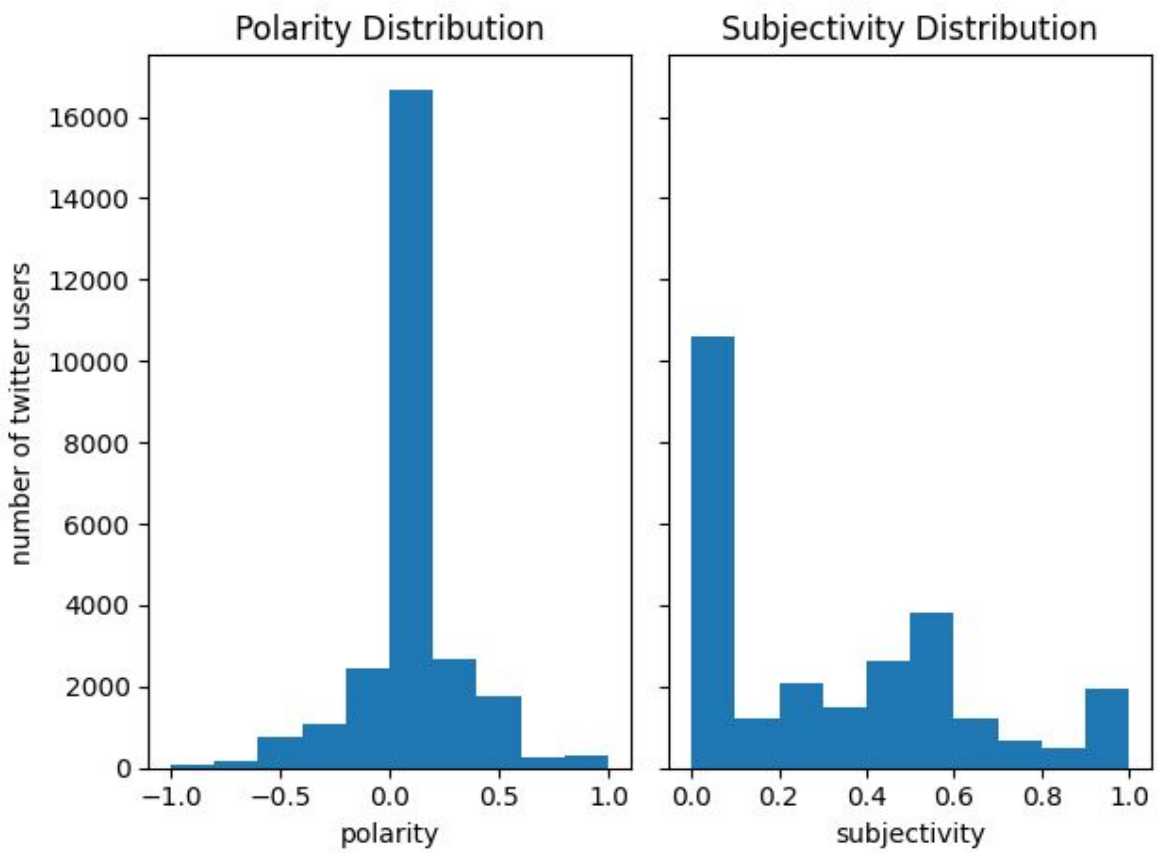

polarity vs subjectivity
polarity
subjectivity
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0.0
0.2
0.4
0.6
0.8
1.0


Polarity Distribution
Subjectivity Distribution
number of twitter users
3000
2500
2000
1500
1000
500
0
−1.0
−0.5
0.0
0.5
1.0
polarity
0.0
0.2
0.4
0.6
0.8
1.0
subjectivity

2020 Election:







Next, the results for k nearest neighbor classification using five features to predict subjectivity or polarity.

Data privacy(subjectivity):

```
Run: KNN
C:\Users\owner\Twitter_Project\Scripts\python.exe C:/Users/owner/PycharmProjects/Twitter_Project/KNN.py
[[491  25   9]
 [ 72 120  20]
 [ 63  28  71]]
              precision    recall  f1-score   support

           0       0.78      0.94      0.85       525
           1       0.69      0.57      0.62       212
           2       0.71      0.44      0.54       162

    accuracy                           0.76       899
   macro avg       0.73      0.65      0.67       899
weighted avg       0.75      0.76      0.74       899


Process finished with exit code 0
```

Data privacy(polarity):

```
Run: KNN
C:\Users\owner\Twitter_Project\Scripts\python.exe C:/Users/owner/PycharmProjects/Twitter_Project/KNN.py
[[  6  25  15]
 [  9 545  67]
 [  8  90 134]]
              precision    recall  f1-score   support

           0       0.26      0.13      0.17        46
           1       0.83      0.88      0.85       621
           2       0.62      0.58      0.60       232

    accuracy                           0.76       899
   macro avg       0.57      0.53      0.54       899
weighted avg       0.74      0.76      0.75       899


Process finished with exit code 0
```

2020 Voting Election(subjectivity):

```
Run: KNN
C:\Users\owner\Twitter_Project\Scripts\python.exe C:/Users/owner/PycharmProjects/Twitter_Project/KNN.py
[[3001  145  104]
 [ 374  521  121]
 [ 421  163  393]]
              precision    recall  f1-score   support

           0       0.79      0.92      0.85      3250
           1       0.63      0.51      0.56      1016
           2       0.64      0.40      0.49       977

    accuracy                           0.75      5243
   macro avg       0.68      0.61      0.64      5243
weighted avg       0.73      0.75      0.73      5243


Process finished with exit code 0
```

2020 Voting Election(polarity):

```
Run: KNN
C:\Users\owner\Twitter_Project\Scripts\python.exe C:/Users/owner/PycharmProjects/Twitter_Project/KNN.py
[[  2  26  21]
 [  8 549  70]
 [ 13  81 129]]
              precision    recall  f1-score   support

           0       0.09      0.04      0.06        49
           1       0.84      0.88      0.86       627
           2       0.59      0.58      0.58       223

    accuracy                           0.76       899
   macro avg       0.50      0.50      0.50       899
weighted avg       0.73      0.76      0.74       899


Process finished with exit code 0
```

Based on the results above, we can conclude that it is possible to predict polarity and subjectivity. Since, the accuracy is 75 to 76 for both polarity and subjectivity.