Social Media Sentiment Analytics using Python, R Programming and K-Means Clustering Algorithm

K-Means Clustering

(MacQueen 1967) is one of the most commonly used **unsupervised** machine learning algorithm for partitioning a given data set into a set of k groups (i.e. *k clusters*), where k represents the number of groups pre-specified by the analyst. It **classifies objects in multiple groups** (i.e., clusters), such that objects within the same cluster are as similar as possible (i.e., high *intra-class similarity*), whereas objects from different clusters are as dissimilar as possible (i.e., low *inter-class similarity*). In k-means clustering, **each cluster is represented by its center** (i.e, *centroid*) which corresponds to the mean of points assigned to the cluster.

Algorithm

STEP 1 : Specify the number of clusters (K) to be created.
STEP 2 : Select randomly k objects from the dataset as the initial cluster centers or means
STEP 3 : Assigns each observation to their closest centroid, based on the Euclidean distance between the object and the centroid.
STEP 4 : For each of the k clusters update the *cluster centroid* by calculating the new mean values of all the data points in the cluster. The centoid of a *K**t**h* cluster is a vector of length *p* containing the means of all variables for the observations in the *k**t**h* cluster; *p* is the number of variables.
STEP 5 : Iteratively minimize the total within sum of square. That is, iterate steps 3 and 4 until the cluster assignments stop changing or the maximum number of iterations is reached. By default, the **R** software uses 10 as the default value for the maximum number of iterations.

Extracting Twitter Data using Python

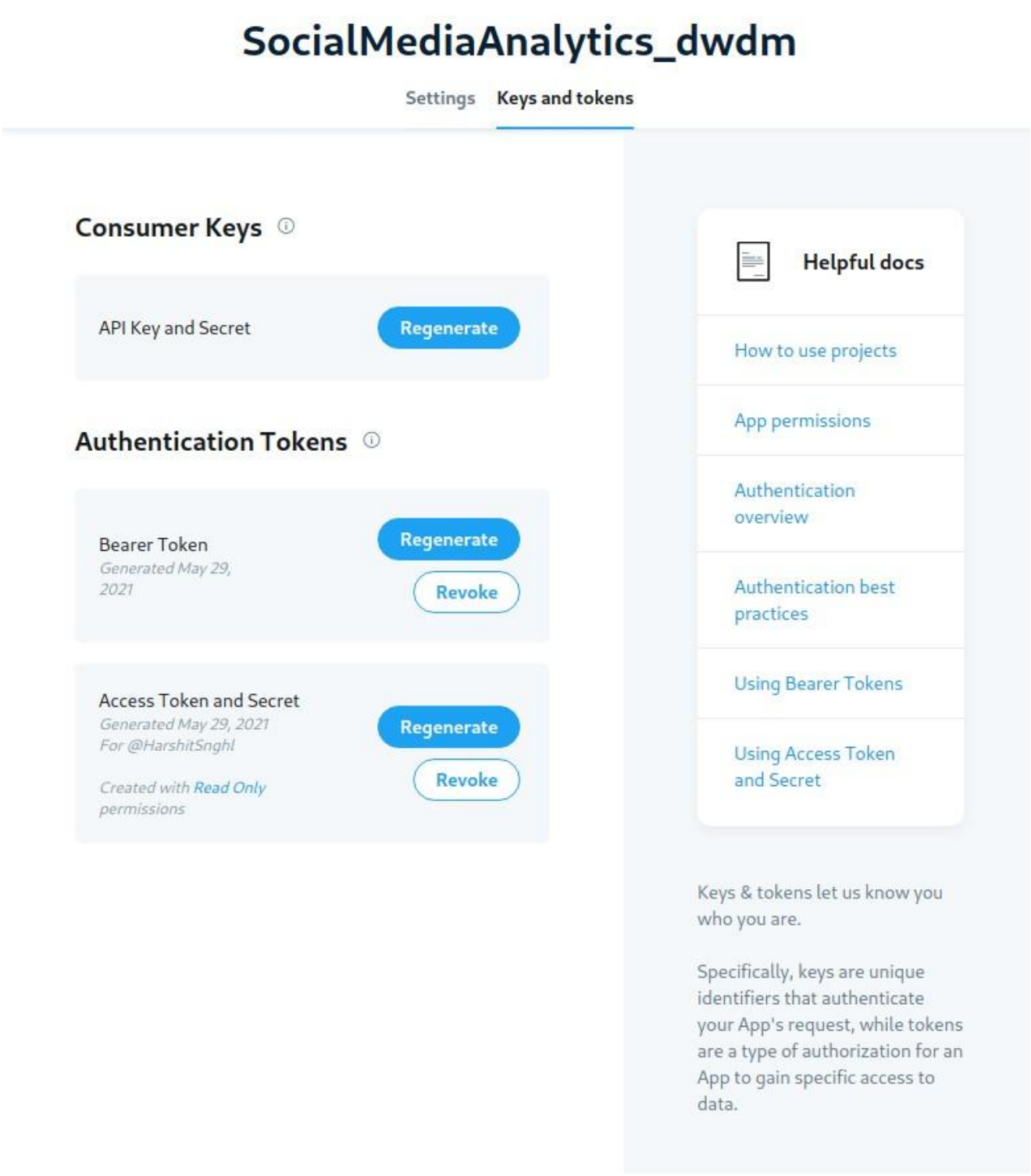
STEP 1 : Configuring Python Environment

- Install `tweepy` & `textblob` package for python using the following commands

```
pip install tweepy
pip install textblob
```

STEP 2 : Generating authorisation key to extract twitter data

- We can generate authorization keys by creating an app for Twitter in Twitter Developer Portal



- Put the authoristion keys in a file called auth.k as given below

```
CONSUMER_KEY OR API_KEY
CONSUMER_SECRET OR API_SECRET
ACCESS_TOKEN OR AUTHORISATION_TOKEN
ACCESS_TOKEN_SECRET OR AUTHORISATION_SECRET
```

STEP 3 : Create a python script names twitter_search.py as follows

```
    #!/usr/bin/env python3
    # -*- coding: utf-8 -*-
    """
    Created on Sun May 30 02:26:14 2021

    @author: harshit
    """

    # twitter_sentiment_search.py

    from textblob import TextBlob
    import csv
    import tweepy
    import unidecode

    # AUTHENTICATION (OAuth)
    f = open('auth.k', 'r')
    ak = f.readlines()
    f.close()
    auth1 = tweepy.auth.OAuthHandler(
        ak[0].replace("\n", ""), ak[1].replace("\n", ""))
    auth1.set_access_token(ak[2].replace("\n", ""), ak[3].replace("\n", ""))
    api = tweepy.API(auth1)

    # Tweeter search with keyword
    target_num = 1125458
    # No. of tweets to be fetched. The fetched tweets will be less than this
number because of the restriction of the twitter api to fetch tweets per
minute by a partucular api key
    query = "elon"

    csvFile = open('elon_sentiment_result.csv', 'w')
    csvWriter = csv.writer(csvFile)
    csvWriter.writerow(["username", "author id", "created", "text", "retwc",
"hashtag",
                        "followers", "friends", "favorite_count", "polarity",
"subjectivity"])
    counter = 0

    for tweet in tweepy.Cursor(api.search, q=query, lang="en",
result_type="mixed", count=target_num).items():
        created = tweet.created_at
```

```
        text = tweet.text
        text = unidecode.unidecode(text)
        retwc = tweet.retweet_count
        favorite_count = tweet.favorite_count
        try:
            hashtag = tweet.entities[u'hashtags'][0][u'text']  # hashtags
used
        except:
            hashtag = "None"
        username = tweet.author.name  # author/user name
        authorid = tweet.author.id  # author/user ID#
        # number of author/user followers (inlink)
        followers = tweet.author.followers_count
        # number of author/user friends (outlink)
        friends = tweet.author.friends_count
        text_blob = TextBlob(text)
        polarity = text_blob.polarity
        subjectivity = text_blob.subjectivity
        csvWriter.writerow([username, authorid, created, text, retwc,
                            hashtag , followers, friends, favorite_count,
polarity, subjectivity])

        counter = counter + 1
        print(counter)
        if (counter == target_num):
            break

    csvFile.close()
```

STEP 4 : We will have two scripts that will help us to get the twitter data

- auth.k & twitter_search.py
- The ‘auth.k’ file will contain the authentication information as shown above so that ‘twitter_sentiment_search.py’ could access the twitter data from the server.

  Place auth.k in the same directory where twitter_search.py locates.

```
CONSUMER_KEY OR API_KEY
CONSUMER_SECRET OR API_SECRET
ACCESS_TOKEN OR AUTHORISATION_TOKEN
ACCESS_TOKEN_SECRET OR AUTHORISATION_SECRET
```

- *Note: The order of the keys must be the same as shown above*

- The above scripts will generate a file names `elon_sentiment_result.csv` containing the data extracted from the twitter with polarity and subjectivity.
- Polarity and Subjectivity
- `Polarity` Positive, negative, and neutral (+, -, 0). E.g. product reviews and movie reviews. (numerical)
  `Subjectivity` → Text classified into one of two classes: objective and subjective (0 to 1). Depends on context, category, and criteria of subjectivity. (categorical) **STEP 1** Executing the scripts

```
python twitter_sentiment_search .py
```

Note : One can change the query parameter in the script to fetch the result of their result, the script could exit with a error code but it is nothing to worry about

- Snapshot of the extracted data is as follows

`elon_sentiment_result.csv`

author.id	created	text	retwc	hashtag	followers	friends	favorite_count	polarity	subjectivity
1.392445e+18	2021-06-03 19:18:41	RT @daulani08: Daily Dose: Elon Musk- The Man wh...	4	معصيتي_راحتي	14	29	0	0.000000000	0.00000000
1.391494e+18	2021-06-03 19:17:29	RT @daulani08: Daily Dose: Elon Musk- The Man wh...	4	معصيتي_راحتي	15	74	0	0.000000000	0.00000000
1.392050e+18	2021-06-03 19:16:13	RT @daulani08: Daily Dose: Elon Musk- The Man wh...	4	معصيتي_راحتي	14	65	0	0.000000000	0.00000000
1.392810e+18	2021-06-03 19:08:38	Daily Dose: Elon Musk- The Man who never give up....	4	معصيتي_راحتي	24	88	3	0.000000000	0.00000000
1.370101e+18	2021-06-04 00:56:51	RT @Autogrillocryp: This is incredible! Did you see ...	19	ZABAKU	41	267	0	1.000000000	0.95000000
9.199054e+08	2021-06-03 19:31:41	RT @Autogrillocryp: This is incredible! Did you see ...	19	ZABAKU	29	198	0	1.000000000	0.95000000
1.284691e+18	2021-06-03 17:16:10	RT @Autogrillocryp: This is incredible! Did you see ...	19	ZABAKU	256	415	0	1.000000000	0.95000000
1.365946e+18	2021-06-03 14:08:21	RT @YouSwap_Global: ATTENTION! #YouSwap #Giv...	6784	YouSwap	22	360	0	0.000000000	0.00000000
9.943415e+17	2021-06-03 18:38:20	@jacksfilms #YIAYjoker Elon musk stole my idea for ...	0	YIAYjoker	6	8	0	0.000000000	0.00000000
1.057782e+18	2021-06-03 17:23:30	@jacksfilms #YIAYe3 Excited Elon Ejaculation	0	YIAYe3	69	2334	0	0.375000000	0.75000000
1.365919e+18	2021-06-03 17:15:44	@jacksfilms #YIAYe3 Elon Eats Ebola. 'nuff said.	0	YIAYe3	0	8	0	0.000000000	0.00000000
2.465724e+08	2021-06-03 15:47:51	@jacksfilms Elon Eliminates E-coin Take that liberal...	0	YIAYe3	24	44	0	0.000000000	0.00000000
1.194280e+18	2021-06-03 13:38:11	@jacksfilms #YIAYe3 e3 stands for, Elon, Elon, Elon, ...	0	YIAYe3	10	61	1	0.000000000	0.00000000
9.551661e+17	2021-06-03 16:40:21	@SpaceX @elonmusk @NASA #xvs coin up up up ;)...	0	xvs	23	78	1	0.225000000	0.60000000
3.389049e+09	2021-06-03 21:54:39	RT @RuleXRP: #xrpcommunity Jay Clayton, the cro...	33	xrpcommunity	75	334	0	0.061904762	0.37857143
1.127870e+18	2021-06-03 19:52:53	RT @RuleXRP: #xrpcommunity Jay Clayton, the cro...	33	xrpcommunity	56	360	0	0.061904762	0.37857143
1.149601e+18	2021-06-03 19:51:11	RT @RuleXRP: #xrpcommunity Jay Clayton, the cro...	33	xrpcommunity	6576	734	0	0.061904762	0.37857143
9.458363e+17	2021-06-03 16:56:45	RT @RuleXRP: #xrpcommunity Jay Clayton, the cro...	33	xrpcommunity	51	352	0	0.061904762	0.37857143
1.347444e+18	2021-06-03 15:33:53	RT @RuleXRP: #xrpcommunity I do not follow Elon ...	8	xrpcommunity	129	463	0	0.000000000	0.50000000
5.531514e+08	2021-06-03 15:03:40	RT @RuleXRP: #xrpcommunity I do not follow Elon ...	8	xrpcommunity	1010	1362	0	0.000000000	0.50000000
1.264634e+18	2021-06-03 14:58:39	RT @RuleXRP: #xrpcommunity I do not follow Elon ...	8	xrpcommunity	321	349	0	0.000000000	0.50000000
1.018819e+18	2021-06-03 14:53:21	RT @RuleXRP: #xrpcommunity I do not follow Elon ...	8	xrpcommunity	99	402	0	0.000000000	0.50000000
3.245376e+09	2021-06-03 14:46:45	RT @RuleXRP: #xrpcommunity I do not follow Elon ...	8	xrpcommunity	216	595	0	0.000000000	0.50000000
2.756194e+09	2021-06-03 14:45:41	RT @RuleXRP: #xrpcommunity I do not follow Elon ...	8	xrpcommunity	825	293	0	0.000000000	0.50000000

Implementing K-Means Clustering Algorithm

STEP 1 : Installing R packages in RStudio environment

- ggpubr: creates plots.

```
install.packages("ggpubr", dependencies = TRUE, repos = "http://cran.us.r-project.org")
```

- factoextra: Extract and Visualize the Results of Multivariate Data Analyses.

```
install.packages("factoextra", dependencies = TRUE, repos =
"http://cran.us.r-project.org" )
```

STEP 2 : Importing required packages

```
library(ggpubr)
library(factoextra)
```

STEP 3 : Loading the data

```
# Select the elon_sentiment_result.csv file
twitter_sentiment_data  = read.csv(file.choose(),header=TRUE,sep=',')
head(twitter_sentiment_data ,2)
```

```
>
> head(twitter_sentiment_data,2)
                username    author.id            created
1 Lifui. O. Lor. 🥤🚀🎯 (Ace) 9.630489e+17 2021-06-06 08:51:55
2            miguel ⏏ BLM! 9.419189e+17 2021-06-06 08:51:55
                                                                                                                                                text
1 RT @BakingBad9: @elonmusk Save my portfolio Elon. My wife is threatening to leave me, I put our money we saved for a house in $DOGE and wen...
2 @christianboedya @hottienashe @th3mb0fication azealia and grimes were going to collab, so azealia got flewed out an... https://t.co/RbSBL8ESaz
  retwc hashtag followers friends favorite_count polarity subjectivity sentiment
1    19    None       897      14              0        0           0   Neutral
2     0    None      1787    1507              0        0           0   Neutral
> |
```

STEP 4 : Cleaning the data

- Choosing the 10th and 11th column.

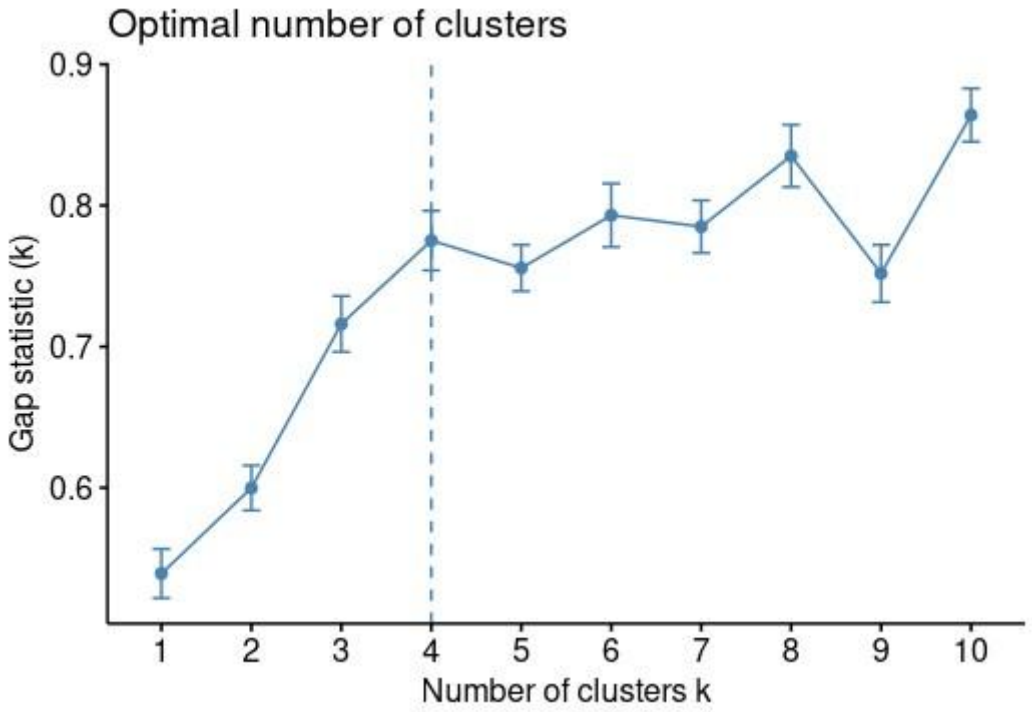
```
twitter_sentiment_data  = subset(twitter_sentiment_data ,
(twitter_sentiment_data $retwc > 0 & twitter_sentiment_data $followers>0 &
twitter_sentiment_data $friends>0 & twitter_sentiment_data $favorite_count >0
))

tws_data = twitter_sentiment_data [,c(10,11)]
```

	polarity	subjectivity
77	0.17500000	0.52500000
222	0.00000000	0.0000 0.5250000
312	0.00000000	0.00000000
332	0.00000000	0.00000000
439	0.00000000	0.00000000
460	0.00000000	0.00000000
570	0.00000000	0.00000000
655	0.00000000	0.00000000
662	0.10000000	1.00000000
663	0.00000000	0.33333333
665	0.80000000	0.40000000
686	0.35000000	0.50000000
753	0.10000000	1.00000000
794	-0.25000000	0.40000000
816	0.25000000	0.50000000
819	0.60000000	0.90000000
833	0.00000000	0.00000000
867	0.00000000	0.00000000
887	0.00000000	0.00000000
902	0.00000000	0.00000000
911	0.37500000	0.50000000
947	0.10000000	1.00000000
959	0.00000000	0.00000000
971	0.70000000	0.60000000
996	0.00000000	0.00000000
1050	-0.17857143	0.28571429
1074	0.00000000	0.00000000
1114	0.25000000	0.33333333

Showing 1 to 29 of 402 entries, 2 total columns

STEP 5 : Determining the optimal number of clusters

```
#Determining The Optimum Number Of Clusters
fviz_nbclust(tws_data, kmeans, method = "gap_stat")
```



Note : This will take some time depending on the hardware if the dataset is too large

- The idea is to compute k-means clustering using different values of clusters k. Next, the WSS (Within Sum of Square) is drawn according to the number of clusters. The location of a bend (knee) in the plot is generally considered as an indicator of the appropriate number of clusters which in this case seems to be k = 4

STEP 6 : Computing k-means clustering

- Calculating k-means clustering using **k = 4**. As the final result of k-means clustering result is sensitive to the random starting assignments, we specify **nstart = 402**. This means that R will try 402 different random starting assignments and then select the best results. Here 402 is our no. of observations.

- As k-means clustering algorithm starts with k randomly selected centroids, it's always recommended to use the ***set.seed()*** function in order to set a seed for *R's random number generator*. The aim is to make reproducible the results, so that the reader of this article will obtain exactly the same results as those shown below.
- Scaling the data using **scale()** to make variables comparable.

```
set.seed(123)
kmeans.res <- kmeans(scale(tws_data), center = 4, nstart = 402)
summary(kmeans.res)
print(kmeans.res)
```

```
>
> set.seed(123)
> kmeans.res <- kmeans(scale(tws_data), center = 4, nstart = 25)
> summary(kmeans.res)
             Length Class  Mode
cluster      17672  -none- numeric
centers          8  -none- numeric
totss            1  -none- numeric
withinss         4  -none- numeric
tot.withinss     1  -none- numeric
betweenss        1  -none- numeric
size             4  -none- numeric
iter             1  -none- numeric
ifault           1  -none- numeric
> print(kmeans.res)
K-means clustering with 4 clusters of sizes 8697, 5683, 1441, 1851

Cluster means:
     polarity subjectivity
1 -0.01706447   -0.8744120
2 -0.02249185    0.6195375
3 -2.23918112    1.4315153
4  1.89243161    1.0919051

Clustering vector:
   [1] 1 1 4 2 2 2 2 2 2 1 3 2 1 4 2 1 4 1 1 2 2 2 4 1 2 2 2 1 1 2 2 2 2 2 1 1 3
  [38] 2 2 2 2 2 1 1 2 2 2 1 2 2 1 1 2 1 2 1 2 2 1 2 2 1 2 1 2 2 2 2 1 1 3 1 2 1
  [75] 2 4 2 1 1 2 3 1 1 2 4 1 2 2 3 1 3 1 4 2 1 2 2 2 1 1 1 1 1 1 2 1 2 2 3 1 1
 [112] 3 1 1 2 4 1 1 1 1 1 2 2 2 2 2 2 1 1 1 2 2 1 1 2 1 2 1 4 3 1 1 2 2 2 1 4 1
 [149] 1 1 1 1 4 1 2 1 4 2 1 2 1 1 3 1 1 4 2 1 2 2 1 2 1 1 2 4 1 1 2 3 3 1 3 2 1
 [186] 2 4 2 1 1 1 1 4 1 1 1 1 1 4 2 1 1 1 1 2 1 1 3 2 1 2 2 1 2 1 1 1 2 2 2 2 1
 [223] 1 1 2 4 1 1 3 2 1 1 1 3 1 1 4 2 2 1 1 1 3 1 2 1 1 1 1 2 1 1 2 1 2 4 2 1 1
 [260] 2 1 2 1 1 2 1 2 2 2 1 2 2 3 1 1 1 1 1 2 1 2 1 1 2 4 2 1 1 1 2 1 2 1 1 1 3
 [297] 1 4 2 1 1 2 2 4 2 1 1 1 4 1 1 1 1 1 1 2 4 2 4 2 1 1 1 2 1 2 1 1 2 3 1 1 1
 [334] 2 2 2 1 2 2 1 1 1 2 1 1 4 2 1 2 2 4 1 3 1 3 1 1 4 1 1 2 1 1 1 1 2 1 1 1 1
 [371] 1 1 2 3 1 1 1 2 1 1 2 1 1 2 1 3 1 1 1 1 4 1 4 2 1 1 1 2 1 1 2 1 2 1 1 1 3
 [408] 1 2 1 2 2 1 1 2 4 1 1 1 1 1 1 1 1 3 1 2 1 1 3 2 4 2 1 2 1 2 2 1 1 2 4 4 4
 [445] 3 2 1 2 1 2 4 1 3 3 2 1 4 1 2 1 2 1 1 2 3 2 4 1 2 3 4 1 2 1 2 2 3 2 1 1 2
 [482] 1 2 1 4 1 1 1 3 1 1 2 2 2 1 2 1 4 2 1 2 1 1 1 1 3 1 2 2 3 1 4 4 2 2 1 1 4
 [519] 1 1 3 1 1 2 2 4 1 1 1 1 2 1 1 1 1 1 2 2 4 2 3 2 2 1 2 4 2 1 4 1 2 2 2 1 1
 [556] 3 3 4 1 1 3 4 3 3 1 1 2 2 1 1 1 1 2 2 4 1 4 4 2 2 3 2 2 2 1 2 3 1 1 4 4 2
 [593] 1 1 1 2 2 1 2 1 1 1 1 2 3 1 1 1 1 1 1 1 2 1 1 4 1 3 1 1 1 2 4 2 4 1 1 4
 [630] 1 2 2 3 2 1 3 2 2 1 1 4 1 4 1 1 1 1 1 1 3 1 1 4 1 1 1 1 4 3 1 1 2 2 1 4 2
 [667] 4 2 2 2 1 1 4 3 1 1 1 1 2 1 1 1 1 2 1 4 1 4 4 3 1 2 3 4 1 2 1 1 2 2 1 2 1
 [704] 1 2 1 4 1 2 3 1 4 2 1 1 1 1 3 1 2 2 1 2 1 1 1 2 2 1 1 1 2 4 2 2 1 2 1 1 1
 [741] 4 2 2 4 2 1 1 1 1 1 1 1 2 1 1 1 1 2 2 1 4 1 3 1 1 3 4 1 2 2 4 2 1 2 4 2 2
 [778] 4 4 3 1 3 1 1 4 3 4 1 4 2 1 1 1 2 2 1 2 1 4 1 1 4 1 2 2 1 4 2 1 1 4 1 4 1
 [815] 2 2 1 1 4 2 1 4 1 2 1 1 4 1 1 3 1 3 1 1 1 1 2 2 1 4 4 4 2 1 1 1 3 1 2 2 1
 [852] 1 3 1 3 1 1 3 3 1 1 4 4 2 1 4 1 2 1 1 1 4 2 3 1 2 4 1 4 1 2 1 1 3 1 2 1 3
 [889] 1 1 2 1 3 1 1 2 1 1 1 4 1 1 1 2 4 1 1 1 1 1 4 4 4 1 3 4 1 1 1 4 4 1 4 1 1
 [926] 1 2 1 2 1 1 1 2 1 1 3 2 1 1 3 1 3 1 1 4 1 2 2 1 1 3 1 1 1 1 4 1 1 1 2 1 1
 [963] 1 4 1 1 4 1 3 1 4 1 1 2 1 2 1 4 2 1 2 2 1 1 1 4 4 1 3 4 2 2 2 3 1 1 1 2 2
[1000] 1
 [ reached getOption("max.print") -- omitted 16672 entries ]

Within cluster sum of squares by cluster:
[1] 1351.326 3548.806 1014.069 1577.535
 (between_SS / total_SS =  78.8 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"
> |
```

- **kmeans()** function returns a list of components, including

  Cluster: A vector of integers (from 1:k) indicating the cluster to which each point is allocated

  Centers: A matrix of cluster centers (cluster means)

  totss: The total sum of squares (TSS□, i.e ∑(xi−x¯)2∑(xi−x¯)2. TSS measures the total variance in the data.

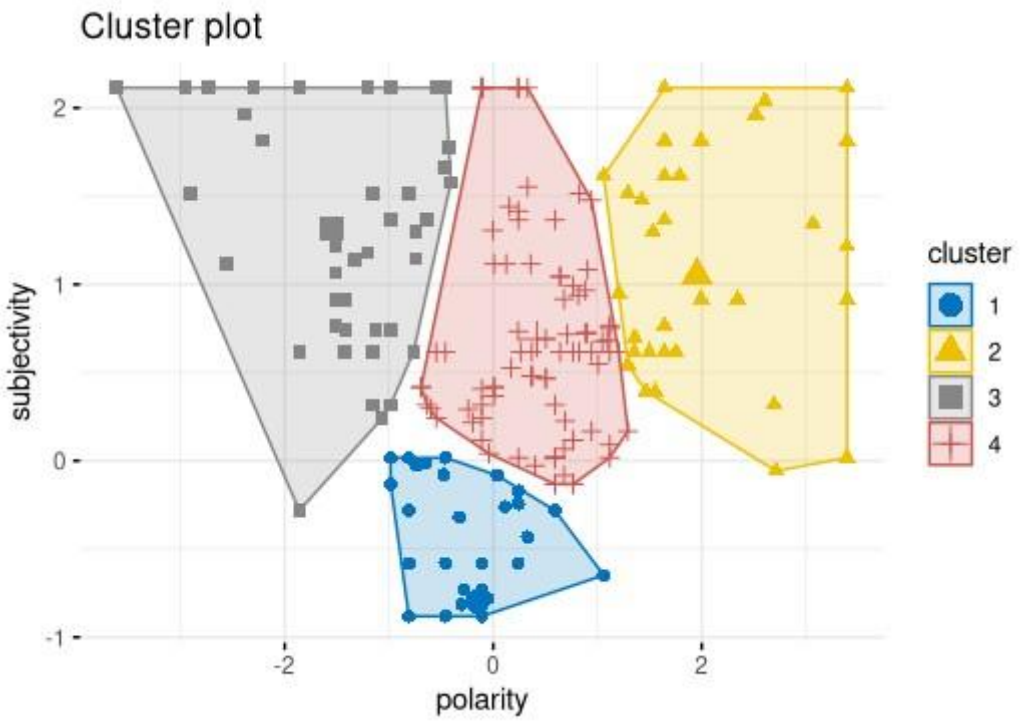
withinss: Vector of within-cluster sum of squares, one component per cluster **withinss**: Total within-cluster sum of squares, i.e. sum(withinss)sum(withinss) **betweenss**: The between-cluster sum of squares, i.e. totss−tot.withinsstotss−tot.withinss

size: The number of observations in each cluster

STEP 6: Visualising Data

- Next we move on to visualizing the clusters for which we shall use the **fviz_cluster()** function.
- The problem is that the data contains more than 2 variables and the question is what variables to choose for the xy scatter plot.
- A solution is to reduce the number of dimensions by applying a dimensionality reduction algorithm, such as Principal Component Analysis (PCA, that operates on the four variables and outputs two new variables (that represent the original variables) that we can use to do the plot.

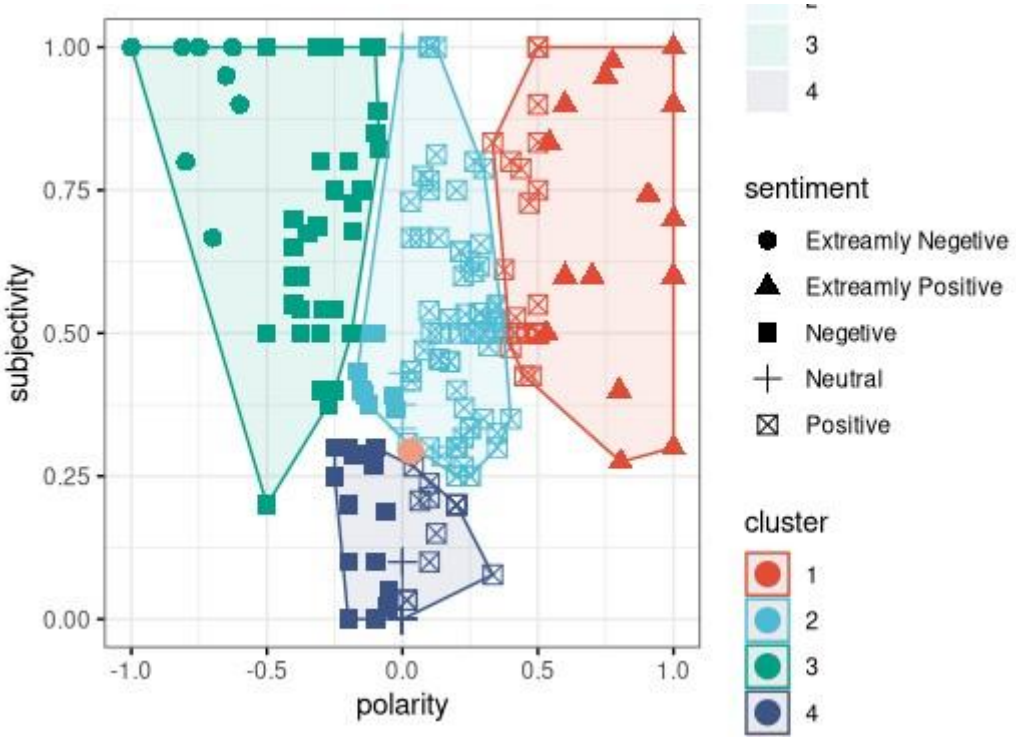
```
fviz_cluster(kmeans.res, data = tws_data,
     ellipse.type = "convex",
     palette = "jco",
     ggtheme = theme_minimal(),
     outlier.color = "black",
     outlier.shape = "*",
     labelsize = 0,
     pointsize = 2,
)
```



STEP 7 : Visualising clusters categorically

- Indentifying nodes with their respective sentiment.

```
# Coordinates of individuals
ind.coord <- tws_data

# Add clusters obtained using the K-means algorithm
ind.coord$cluster <- factor(kmeans.res$cluster)

# Add sentiment groups from the original data sett
ind.coord$sentiment <- twitter_sentiment_data $sentiment


ggscatter(
  ind.coord, x = "polarity", y = "followers",
  color = "cluster", palette = "npg", ellipse = TRUE, ellipse.type =
"convex",
  shape = "sentiment", size = 3,  legend = "right", ggtheme = theme_bw(),
) + stat_mean(aes(color = 'cluster'), size = 4)
```



Conclusion for Subjectivity v/s Polarity clusters • It can be clearly observed that the tweets which are Extreamly Negative and Extreamly Positive are highlysubjective in nature rather than objective which concludes the fact that extreame sentiments are subjective and are clustered in green and red cluster respectively.

- It can be observed that the tweets have a equal distribution of Negative, Extreamly Negative, Extreamly Positive and Positive tweets.
- No. of neutral tweets are less as compared to other categories.