**Codes Snippets:**

#Loading the required Packages

library(plyr)

library(dplyr) #Data Manipulation

library(readr) #String Manipulation

library(stringr) #String Manipulation

library(tm) #Text Mining

library(ggplot2) #Data Visualisation

library(wordcloud) #Word Cloud

library(rlang)

library(qdap) #Text Mining

library(twitteR) #Extracting Twitter Data

library(RSentiment) #Sentiment Analysis

library(knitr)

library(lubridate)

library(scales)

library(syuzhet)

library(reshape2)

library(RColorBrewer)# Install packages for sentiment analysis

library("gridExtra")

#used on LDA

library(topicmodels)

library(quanteda)

options(warn=-1) #Suppress Warnings

#Data Processing

#We have already harvested the tweets and fetched texts from the tweets into text file: modi.csv.

#So, will skip the initial part of coding showing fetching tweets.

#Rather we will start by reading the already saved file and then will show the data cleaning and processing step by step.

df=read.csv("E:/Practicum/Modi/Github/demonetization-tweets.csv")

#Before going into the data cleaning step couple of things are to be cleared:

#a) It's very important to maintain logical order in executing the cleaning commands. Other wise some information can be missed unintentionally.

#For example, if we convert all the tweets and later on apply 'gsub' command to remove retweets with 'rt' pattern we may lose part of words

#that contain 'rt'.

#b) Throughout the cleaning step it's a good practice to randomly check the csv file to make sure no unexpected transformation takes place.

#For example, I will view 1500th tweet from my file as a benchmark. That tweet I picked arbitrarilly.

#I will check text of that tweet before starting the data cleaning process and also will view at different points during the cleaning steps.

df$text[1500]

#In the following section I start data cleaning process by converting the text to ASCII format to get rid of the funny characters usually used in

#Twitter messages.

df$text<- iconv(df$text, to = "ASCII", sub = " ")

# Data Preprocessing of Tweets

#On the following code section, I will apply bunch of codes to remove special characters, hyperlink, usernames, tabs, punctuations and unnecessary

#white spaces. Have mentioned specific use of each code along with the codes below.

#We will remove hashtags, junk characters, other twitter handles and

#URLs from the tags using gsub function so we have tweets for further analysis

df$text=gsub("&amp", "", df$text)

#Cleaning up twitter handles

df$text = gsub("(RT|via)((?:\\b\\W\*@\\w+)+)", "", df$text) #Removes the "RT" retweet and usernames

#Removing User Names

df$text = gsub("@\\w+", "", df$text)

#Cleaning up punctuations and digits

df$text = gsub("[[:punct:]]", "", df$text) #removes punctuations

df$text = gsub("[[:digit:]]", "", df$text) #removes digits

#Cleaning up URL's

df$text = gsub("http\\w+", "", df$text) #removes http links

df$text = tolower(df$text) #converting text to lower case

#Convert the timestamps of the tweets to the same time zone

#Change the time zone to Asia/Kolkata

#The extracted tweets have a field created that shows the date timestamp of tweets in UTC time zone

df$created <- ymd\_hms(df$created)

df$created <- with\_tz(df$created, "Asia/Kolkata")

#Show twitter volume by weekdays and weekends

ggplot(data = df, aes(x = wday(created, label = TRUE))) +

geom\_bar(aes(fill = ..count..)) +

theme(legend.position = "none") +

xlab("Day of the Week") + ylab("Number of tweets") +

scale\_fill\_gradient(low = "midnightblue", high = "aquamarine4")

# We need the text column for the sentiment analysis

text=as.character(df$text)

sampleData=sample(text,length(text))

# Lets Create a corpus which is a collection of texts

#As the next step of data processing I will convert this tweets file, which is a character vector, into a corpus.

#In general term, corpus in linguistic means a structured set of texts that can be used for statistical analysis, hypothesis testing,

#occurance checking and validating linguistic rules.

#To achive this goal I will use 'corpus' and 'VectorSource' commands from 'tm' library in R.

#While 'VectorSource' will interpret each element of our character vector file 'tweets' as a document and feed that input into 'corpus' command.

#Which eventually will convert that into corpus suitable for statistical analysis.

corpus=Corpus(VectorSource(list(sampleData)))

head(corpus)

# Lets remove the unwanted texts in the corpus

#more cleaning on the corpus by removing stop words and numbers because both these have very little value, towards our goal of sentiment analylsis and topic modeling.

#At this step I will convert the words in the corpus into stem words. In general terms, word stemming means the process of reducing a word to its base form

#which may or may not be the same as the morphological root of the word or may or may not bear meaning by the stem word itself.

#In addition to the stemming I will also delete my search key 'Modi' from the tweets and amp'is abbrebiation of 'Accelerated Mobile Page'

corpus <- tm\_map(corpus, content\_transformer(replace\_abbreviation))

corpus <- tm\_map(corpus, removePunctuation)

corpus <- tm\_map(corpus, removeNumbers)

corpus <- tm\_map(corpus, content\_transformer(tolower))

corpus <- tm\_map(corpus, content\_transformer(bracketX))

corpus <- tm\_map(corpus, removeWords, c("modi","amp", "added","nmapp","two","modi","dhan","jan","terrorism", "post","across","like","will","also", "said","see","end","since","read","share","meet","dont","just","via","per","etc","yes", "no", "well", "eight", "twenty", "now", "can", "even", "still",stopwords("english")))

corpus <- tm\_map(corpus, stripWhitespace)

corpus <- tm\_map(corpus,removeWords,stopwords('english'))

corpus <- tm\_map(corpus,stemDocument)

#Word Cloud : Textual Analysis

#Creating a Document Term Matrix

#convert our corpus in a Document Term Matrix (DTM). DTM creates a matrix that consists all words or terms as an individual column and each document,

#in our case each tweet, as a row. Numeric value of 1 is assigned to the words that apprear in the document from the corresponding row and value of

#0 is assigned to the rest of the words in that row. Thus the resulting DTM file is a sparse which is a large matrix containing a lot of 0.

document=DocumentTermMatrix(VCorpus(VectorSource(corpus[[1]]$content)))

#Creating a Frequent Term Matrix as we need to get the frequency of words

freq=colSums(as.matrix(document))

freq=sort(freq,decreasing=TRUE)

demonetization\_matrix <- as.matrix(document)

freq[1:10]

#I am finally done with our first step of data cleaning and pre-processing. On the next step I will start data processing to create our topic model.

#But before diving into model creation I decided to crate a word cloud to get a feel about the data.

word\_color<-brewer.pal(12,"Paired")

words\_freq <- data.frame(term = names(freq), num = freq)

wordcloud(words\_freq$term, words\_freq$num, colors = word\_color, max.words = 100)

#Topic modeling using LDA

#Have used the 'topicmodel' package available in R for topic modeling. In this model, the number of topics are to be selected.

#It creates the probability of each topic in each document and also distributes the words under each topic.

#Selecting more number of topics may result in more clearer segregation but at the same time the differences among different topics may get blurred.

#While on the other hand selecting very small number of topic can lead to losing possible topic. So to minimze this error , we have tried three

#number of topics to create the LDA model. Used 2, 5 and 10 as the number of topics and created three different LDA models based on these K values.

#LDA model with 5 topics selected

lda\_5 = LDA(document, k = 5, method = 'Gibbs',

control = list(nstart = 5, seed = list(1505,99,36,56,88), best = TRUE,

thin = 500, burnin = 4000, iter = 2000))

#LDA model with 2 topics selected

lda\_2 = LDA(document, k = 2, method = 'Gibbs',

control = list(nstart = 5, seed = list(1505,99,36,56,88), best = TRUE,

thin = 500, burnin = 4000, iter = 2000))

#LDA model with 10 topics selected

lda\_10 = LDA(document, k = 10, method = 'Gibbs',

control = list(nstart = 5, seed = list(1505,99,36,56,88), best = TRUE,

thin = 500, burnin = 4000, iter = 2000))

#Getting the most frequent words under each topic and document wise probability of each topic

#Fetching top 10 terms in each topic:

top10terms\_5 = as.matrix(terms(lda\_5,10))

top10terms\_2 = as.matrix(terms(lda\_2,10))

top10terms\_10 = as.matrix(terms(lda\_10,10))

top10terms\_5

top10terms\_2

top10terms\_10

#We can see that all three models picked the topics of money , deposit , GST and govern.

lda.topics\_5 = as.matrix(topics(lda\_5))

lda.topics\_2 = as.matrix(topics(lda\_2))

lda.topics\_10 = as.matrix(topics(lda\_10))

summary(as.factor(lda.topics\_10[,1]))

topicprob\_5 = as.matrix(lda\_5@gamma)

topicprob\_2 = as.matrix(lda\_2@gamma)

topicprob\_10 = as.matrix(lda\_10@gamma)

#We can also get document wise probability of each topic. Probability of each topic:

head(topicprob\_2,1)#Probability of each topic.

head(topicprob\_5,1)

head(topicprob\_10,1)

#------------------------------------------------------

# Money : Topic Sentiment Analysis

#---------------------------------------------------

corpus\_money = corpus(df$text)

corpus\_money = (corpus\_money= subset(corpus\_money, grepl('money', texts(corpus\_money))))

writeLines(as.character(corpus\_money[[150]]))

#Tokenizing character vector file 'df'.

token\_money = data.frame(text=corpus\_money, stringsAsFactors = FALSE) %>% unnest\_tokens(word, text)

#Matching sentiment words from the 'NRC' sentiment lexicon

senti\_money = inner\_join(token\_money, get\_sentiments("nrc")) %>%

count(sentiment)

senti\_money$percent = (senti\_money$n/sum(senti\_money$n))\*100

#Plotting the sentiment summary

ggplot(senti\_money, aes(sentiment, percent)) +

geom\_bar(aes(fill = sentiment), position = 'dodge', stat = 'identity')+

ggtitle("Money : Sentiment analysis based on lexicon: 'NRC'")+

coord\_flip() +

theme(legend.position = 'none', plot.title = element\_text(size=18, face = 'bold'),

axis.text=element\_text(size=16),

axis.title=element\_text(size=14,face="bold"))

#------------------------------------------------------

# Deposit : Topic Sentiment Analysis

#---------------------------------------------------

corpus\_deposit = corpus(df$text)

corpus\_deposit = (corpus\_deposit = subset(corpus\_deposit, grepl('deposit', texts(corpus\_deposit))))

writeLines(as.character(corpus\_deposit[[150]]))

#Tokenizing character vector file 'df'.

token\_deposit = data.frame(text=corpus\_deposit, stringsAsFactors = FALSE) %>% unnest\_tokens(word, text)

#Matching sentiment words from the 'NRC' sentiment lexicon

senti\_deposit = inner\_join(token\_deposit, get\_sentiments("nrc")) %>%

count(sentiment)

senti\_deposit$percent = (senti\_deposit$n/sum(senti\_deposit$n))\*100

#Plotting the sentiment summary

ggplot(senti\_deposit, aes(sentiment, percent)) +

geom\_bar(aes(fill = sentiment), position = 'dodge', stat = 'identity')+

ggtitle("Deposit : Sentiment analysis based on lexicon: 'NRC'")+

coord\_flip() +

theme(legend.position = 'none', plot.title = element\_text(size=18, face = 'bold'),

axis.text=element\_text(size=16),

axis.title=element\_text(size=14,face="bold"))

#------------------------------------------------------

# Gst : Topic Sentiment Analysis

#---------------------------------------------------

corpus\_gst = corpus(df$text)

corpus\_gst = (corpus\_gst = subset(corpus\_gst, grepl('gst', texts(corpus\_gst))))

writeLines(as.character(corpus\_gst[[150]]))

#Tokenizing character vector file 'df'.

token\_gst = data.frame(text=corpus\_gst, stringsAsFactors = FALSE) %>% unnest\_tokens(word, text)

#Matching sentiment words from the 'NRC' sentiment lexicon

senti\_gst = inner\_join(token\_gst, get\_sentiments("nrc")) %>%

count(sentiment)

senti\_gst$percent = (senti\_gst$n/sum(senti\_gst$n))\*100

#Plotting the sentiment summary

ggplot(senti\_gst, aes(sentiment, percent)) +

geom\_bar(aes(fill = sentiment), position = 'dodge', stat = 'identity')+

ggtitle("Gst : Sentiment analysis based on lexicon: 'NRC'")+

coord\_flip() +

theme(legend.position = 'none', plot.title = element\_text(size=18, face = 'bold'),

axis.text=element\_text(size=16),

axis.title=element\_text(size=14,face="bold"))

#------------------------------------------------------

# Govern : Topic Sentiment Analysis

#---------------------------------------------------

corpus\_govern = corpus(df$text)

corpus\_govern = (corpus\_govern = subset(corpus\_govern, grepl('govern', texts(corpus\_govern))))

writeLines(as.character(corpus\_govern[[150]]))

#Tokenizing character vector file 'df'.

token\_govern = data.frame(text=corpus\_govern, stringsAsFactors = FALSE) %>% unnest\_tokens(word, text)

#Matching sentiment words from the 'NRC' sentiment lexicon

senti\_govern = inner\_join(token\_govern, get\_sentiments("nrc")) %>%

count(sentiment)

senti\_govern$percent = (senti\_govern$n/sum(senti\_govern$n))\*100

#Plotting the sentiment summary

ggplot(senti\_govern, aes(sentiment, percent)) +

geom\_bar(aes(fill = sentiment), position = 'dodge', stat = 'identity')+

ggtitle("Govern : Sentiment analysis based on lexicon: 'NRC'")+

coord\_flip() +

theme(legend.position = 'none', plot.title = element\_text(size=18, face = 'bold'),

axis.text=element\_text(size=16),

axis.title=element\_text(size=14,face="bold"))

#From the resulting word cloud we can see that the words are colored differently, which is based on the frequencies of the words appearing in the tweets.

#Sentiment Analysis

demonetisation\_sentiment <- calculate\_sentiment(names(freq))

ggplot(data=demonetisation\_sentiment,aes(x = sentiment,y = freq)) +

geom\_bar(aes(fill = sentiment), stat = "identity") +

theme(legend.position = "none") +

xlab("Sentiment") + ylab("Score") + ggtitle("Demonetization Data Sentiment Analysis")

#Using RSentiment package lets check the sentiment

sentiments=calculate\_sentiment(names(freq))

sentiment=cbind(sentiments,as.data.frame(freq))

#Identify the positve, negative and Neutral sentiments

positiveSentiments=sentiment[sentiment$sentiment=="Positive",]

negativeSentiments=sentiment[sentiment$sentiment=="Negative",]

neutralSentiments=sentiment[sentiment$sentiment=="Neutral",]

cat("Positive Sentiment Count:", sum(positiveSentiments$freq))

cat("Negative Sentiment Count:", sum(negativeSentiments$freq))

cat("Neutral Sentiment Count:", sum(neutralSentiments$freq))

#Lets have the word Cloud

wordcloud(positiveSentiments$text,positiveSentiments$freq,min.freq = 500,random.order = TRUE,colors = brewer.pal(1,"Dark2"))

wordcloud(negativeSentiments$text,negativeSentiments$freq,min.freq = 500,random.order = TRUE,colors = brewer.pal(1,"Dark2"))

#Generating Sentiment score for each tweet

df$clean\_text <- str\_replace\_all(df$text, "@\\w+", "")

Sentiment <- get\_nrc\_sentiment(df$clean\_text)

alltweets\_senti <- cbind(df, Sentiment)

sentimentTotals <- data.frame(colSums(alltweets\_senti[,c(6:15)]))

names(sentimentTotals) <- "count"

sentimentTotals <- cbind("sentiment" = rownames(sentimentTotals), sentimentTotals)

rownames(sentimentTotals) <- NULL

#Twitter Volume in each category

ggplot(data = sentimentTotals, aes(x = sentiment, y = count)) +

geom\_bar(aes(fill = sentiment), stat = "identity") +

theme(legend.position = "none") +

xlab("Sentiment") + ylab("Total Count") + ggtitle("Total Sentiment Score for All Tweets")

positive<- which(Sentiment$positive >= 3)

df$clean\_text[positive] %>% head(5)

negative<- which(Sentiment$negative >= 3)

df$clean\_text[negative] %>% head(5)

anger<- which(Sentiment$anger >= 3)

df$clean\_text[anger] %>% head(5)

**#For running the shiny codes : Server.R and ui.R , please run the below commands :**

library(shiny)

runGitHub("Text-Analysis-Sentiments","Shreenath117")

**#Scraping Codes : Youtube and NewsAPI**

from selenium import webdriver

from selenium.webdriver.chrome.options import Options

from selenium.webdriver.common.by import By

from selenium.webdriver.support.ui import WebDriverWait

from selenium.webdriver.support import expected\_conditions as EC

import time

import os

import csv

import pandas as pd

# Creates a new .csv file that the data will be written to :

csv\_file = open('C:\\Practicum\\Data source\\Namo\_YT.csv', 'w', encoding="UTF-8", newline="")

writer = csv.writer(csv\_file)

# Defining the variables / headers to be written in the dataframe to which we will be scrapping

writer.writerow(['Link\_URL','Link\_Title','Channel','Views','Uploaded','Comments','Comments\_Author','Comments\_Upvotes','Comments\_Downvotes','Comments\_Posted','Comments\_Replies'])

#Invoking the Web Driver

executable\_path = "C:/chromedriver.exe"

chrome\_options = Options()

#Private browsing

chrome\_options.add\_argument("--incognito")

#Calling a incognito instance to avoid ads coming in the homepage and disturbing the follow of the codes

driver = webdriver.Chrome(executable\_path=executable\_path, chrome\_options=chrome\_options)

driver.get('https://www.youtube.com')#Go to Youtube.com

try:

driver.maximize\_window()#maximize the youtube window

time.sleep(10)# allows the browswer/youtube a load time of 10 seconds.

except:

time.sleep(10)# allows the browswer/youtube a load time of 10 seconds.

try:

search=driver.find\_element\_by\_id("search")

#Using Selenium to type in a Search Term , here we have taken 'Kishore Kumar'

search.send\_keys("Narendra Modi+ english")

button = driver.find\_element\_by\_id("search-icon-legacy")

#Finding the search button

button.click()# clicking the button

time.sleep(10)

#Scraping the Top 10 links that show up in the search results

for i in range(1,21):# picking the 1st 10 URL or links

lists= driver.find\_elements\_by\_xpath("//\*[@id='contents']/ytd-video-renderer["+str(i)+"]//\*[@id='video-title']")# getting the elements by Xpath, which would direct selenium on the course of command further.

for listitem in lists:

scrape\_dict = {}

#Collecting required data on Link Url, Link Title, Subscription Channel, No of Views and time when the video was first uploaded

try:

Link\_URL = listitem.get\_attribute('href')#scraping the URL in range

except:

Link\_URL = ""# giving a null if no URL is found

try:

Link\_Title = listitem.get\_attribute('title')# scraping the titles in range

except:

Link\_Title = ""#giving a null/ no results if no title is found

try:

Channel = listitem.find\_element\_by\_xpath("//\*[@id='contents']/ytd-video-renderer["+str(i)+"]//\*[@id='byline']/a").text# fetching the Channel Name as text

except:

Channel = ""# if no Channel name is found it give a no results

try:

Views = listitem.find\_element\_by\_xpath("//\*[@id='contents']/ytd-video-renderer["+str(i)+"]//\*[@id='metadata-line']/span[1]").text.split()[-2]# scraping the number of views with the given xpath

except:

Views = ""# returns a null if no views value is not found

try:

Uploaded = listitem.find\_element\_by\_xpath("//\*[@id='contents']/ytd-video-renderer["+str(i)+"]//\*[@id='metadata-line']/span[2]").text# giving the variable observations on when the video was uploaded

except:

Uploaded = "" # retunrs a null if uploaded value is missing/not found.

#Printing the link URL's

print("----------------------"+ str(i) +"-----------------------")

print(listitem.get\_attribute('href'))

#Click on the link URL to get into the video and scrape the top 50 comments

listitem.click()

time.sleep(5)#gives 5 seconds load time

#Scroll down the page upto a height given as 300 here

driver.execute\_script("window.scrollTo(0, 300);")

#Giving a time of 10 seconds for the Sort by element to load

time.sleep(10)#gives 10 seconds load time

#Finding the element for clicking on Sort by

Sort\_By = driver.find\_element\_by\_xpath("//\*[@id='icon-label']")

Sort\_By.click()

#Finding the element for clicking on Top Comments

Top\_Comments = driver.find\_element\_by\_xpath("//\*[@id='menu']/a[1]/paper-item/paper-item-body/div[1]")

Top\_Comments.click()

time.sleep(10)

#Scrolling down to load the comments upto 50 to scrape

driver.execute\_script('window.scrollTo(1, 3000);')

time.sleep(10)

driver.execute\_script('window.scrollTo(1, 7000);')

#Looping to scrape top 50 comments sorted as mentioned above

for j in range(1,101):

#For each element, collecting data on who wrote it ( user handle ), when it was posted, how many replies, upvotes it received, downvotes do not have a value - hence not mentioned

try:

Comments = listitem.find\_element\_by\_xpath("//\*[@id='contents']/ytd-comment-thread-renderer["+str(j)+"]//\*[@id='content-text']").text# Fetching the comments with given xpath

except:

Comments = "" #retunrs a null if not found

try:

Comments\_Author = listitem.find\_element\_by\_xpath("//\*[@id='contents']/ytd-comment-thread-renderer["+str(j)+"]//\*[@id='author-text']/span").text#fetching the user handle name as text, with the given xpath

except:

Comments\_Author = "" #returns null if not found

try:

Comments\_Upvotes = listitem.find\_element\_by\_xpath("//\*[@id='contents']/ytd-comment-thread-renderer["+str(j)+"]//\*[@id='vote-count-middle']").text#gets the upvotes count

except:

Comments\_Upvotes = "" #retuns a null if not found

try:

Comments\_Downvotes = listitem.find\_element\_by\_xpath("//\*[@id='contents']/ytd-comment-thread-renderer["+str(j)+"]//\*[@id='button']/yt-icon/")# trys to find the value for downvotes

except:

Comments\_Downvotes ="NA" # retuns NA if not found, there is no downvotes in youtube

try:

Comments\_Posted = listitem.find\_element\_by\_xpath("//\*[@id='contents']/ytd-comment-thread-renderer["+str(j)+"]//\*[@id='published-time-text']/a").text# Scrapping when the comment was posted

except:

Comments\_Posted = "" #returns a null if object is not found

try:

#Here, comments\_replies\_raw takes 2nd last element for the "count" of replies which is a number

Comments\_Replies\_raw = listitem.find\_element\_by\_xpath("//\*[@id='contents']/ytd-comment-thread-renderer["+str(j)+"]//\*[@id='more-text']").text.split()[-2]

#If condition on getting the count of replies , goes to except block if it is 0

if(Comments\_Replies\_raw=="View"):

Comments\_Replies=1

else:

Comments\_Replies = Comments\_Replies\_raw

except:

Comments\_Replies = 0

#Writing all the values to the dictionary with the specific headers as keys.

scrape\_dict['Link\_URL'] = Link\_URL

scrape\_dict['Link\_Title'] = Link\_Title

scrape\_dict['Channel'] = Channel

scrape\_dict['Views'] = Views

scrape\_dict['Uploaded'] = Uploaded

scrape\_dict['Comments'] = Comments

scrape\_dict['Comments\_Author'] = Comments\_Author

scrape\_dict['Comments\_Upvotes'] = Comments\_Upvotes

scrape\_dict['Comments\_Downvotes'] = Comments\_Downvotes

scrape\_dict['Comments\_Posted'] = Comments\_Posted

scrape\_dict['Comments\_Replies'] = Comments\_Replies

#Writing all the values from the dictionary created to the csv file as shown opened above.

writer.writerow(scrape\_dict.values())

#Clicking on the back button to go one step backward in the browser history and looped for each of the top 10 links

driver.back()#goes back to perform the loop

except:

search = ""#if the try is not working it gives ends the process

driver.close()# close the drive i.e. the browsing window.

csv\_file.close()#closing the csv after scrapping data is feed in.

NewsAPI

rom newsapi import NewsApiClient

import pandas as pd

newsapi = NewsApiClient(api\_key='da7c600e44954aacb385c3e44734c0d9')

#Created an API key for Times of India API

#Using that API to create queries as below for all recent articles published about "Business Analytics" from all sources.

#Using get\_everything to get the data required from all sources.

list\_data=[]

for j in range(1,51):

news = newsapi.get\_everything(q='Rafale',

sort\_by='publishedAt',

page=j)

print (news)#prints news artcile extracted from API call,thatcontais 50 articles (the limit with our API key) as shown in the JSON output format

#Default sorting is done by publishedAt, which gives us all the recent articles for Business Analytics.

#Running it in a loop to get >20 values. Max limit =1000 rows

#Range is given till 20 , as the results we get for free are < 1000 , more than 20 gives us the index error.

#Currently in the output we have 981 rows of output with the headers as mentioned from the code below :

#Retrieving the required fields from the JSON output news [ Source ID , Source Name, Author , Title, Description & Content]

for i in range(1,20):

source\_id=news['articles'][i]['source']['id']

source\_name=news['articles'][i]['source']['name']

author=news['articles'][i]['author']

title=news['articles'][i]['title']

description=news['articles'][i]['description']

content=news['articles'][i]['content']

list\_data.append((source\_id,source\_name,author,title,description,content))

news\_dataframe=pd.DataFrame(list\_data,columns=['Source\_id','Source\_name','Author','Title','Description','Content'])

#Storing it in a csv file by opening it and encoded it with utf-8

csv\_file = open('C:\\Users\\karathul\\Desktop\\ISB Materials\\Term - 1\\Data Collection\\newsapi\_MII.csv', 'w', encoding="UTF-8", newline="")

news\_dataframe.to\_csv(csv\_file,encoding='utf-8',index=False)

**References:**

1. **Demonetisation Tweets: Data Source : 8 months : Nov'16 to Jul'17**

[**https://github.com/sourabhdattawad/Demonetization-in-India-Sentiment-Analysis/tree/master/datasets**](https://github.com/sourabhdattawad/Demonetization-in-India-Sentiment-Analysis/tree/master/datasets)

1. [**https://rpubs.com/cosmopolitanvan/timelineanalytics**](https://rpubs.com/cosmopolitanvan/timelineanalytics)
2. [**https://rpubs.com/arafath/twitter\_analysis**](https://rpubs.com/arafath/twitter_analysis)
3. [**https://github.com/mjhea0/twitter-sentiment-analysis**](https://github.com/mjhea0/twitter-sentiment-analysis)
4. **The same scraping Codes have been used here from the subject of Data Collection**
5. **Idea on RShiny :** [**https://github.com/Twitter-Sentiment-Analysis/R**](https://github.com/Twitter-Sentiment-Analysis/R)
6. [**http://www.pmindia.gov.in/en/major-initiatives/**](http://www.pmindia.gov.in/en/major-initiatives/)
7. [**https://rpubs.com/JayEnAar/Demonetisation**](https://rpubs.com/JayEnAar/Demonetisation)
8. **RBI Data set : IDEI :** [**https://github.com/JammiNRao/DeMonetisation/blob/master/RBIPaymentsystemstoMonthAug2018\_cleanedupdata.csv**](https://github.com/JammiNRao/DeMonetisation/blob/master/RBIPaymentsystemstoMonthAug2018_cleanedupdata.csv)
9. [**https://www.thequint.com/news/politics/congress-vs-bjp-2019-social-media-battle**](https://www.thequint.com/news/politics/congress-vs-bjp-2019-social-media-battle)
10. [**https://www.kaggle.com/arathee2/demonetization-in-india-twitter-data/kernels**](https://www.kaggle.com/arathee2/demonetization-in-india-twitter-data/kernels)
11. [**https://economictimes.indiatimes.com/tdmc/your-money/demonetization-anniversary-decoding-the-effects-of-indian-currency-notes-ban/articleshow/61579118.cms**](https://economictimes.indiatimes.com/tdmc/your-money/demonetization-anniversary-decoding-the-effects-of-indian-currency-notes-ban/articleshow/61579118.cms)
12. [**https://www.businesstoday.in/current/economy-politics/top-economists-are-divided-on-demonetisation-who-said-what/story/241412.html**](https://www.businesstoday.in/current/economy-politics/top-economists-are-divided-on-demonetisation-who-said-what/story/241412.html)
13. [**https://www.iimb.ac.in/sites/default/files/2018-06/Demonetisation.pdf**](https://www.iimb.ac.in/sites/default/files/2018-06/Demonetisation.pdf)