Name: Sayyed Sohail Rashid	**Course Name: SMA-LAB**
Class: BE-CO	**Batch: 01**
Roll no: 18CO48	**Experiment No: 05**

Aim: Develop Content (text, emoticons, image, audio, video) based social media analytics model for business.

Code:

```
# -*- coding: utf-8 -*-


!pip install snscrape

# Commented out IPython magic to ensure Python
compatibility.
import pandas as pd
import snscrape.modules.twitter as sntwitter
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
```

```
from nltk.stem.porter import PorterStemmer
import string
import re
import textblob
from textblob import TextBlob
import os
from wordcloud import WordCloud, STOPWORDS
from wordcloud import ImageColorGenerator
import warnings
# %matplotlib inline

os.system("snscrape --jsonl --max-results 5000 --
since 2023-01-31 twitter-search 'Unilever 2023
until:2023-02-07'>text-query-tweets.json")

tweets_df = pd.read_json("text-query-tweets.json"
,lines=True)

tweets_df.tail(5)

tweets_df.to_csv()

df1 = tweets_df[[ 'date', 'rawContent' ,
'renderedContent' , 'user' , 'replyCount'
,'retweetCount' , 'likeCount' , 'lang' , 'place']]

df1.head()
```

```
plt.figure(figsize=(17, 5))
sns.heatmap(df1.isnull(), cbar=True,
yticklabels=False)
plt.xlabel("Column_Name", size=14, weight="bold")
plt.title("Places of missing values in
column",size=17)
plt.show()

import plotly.graph_objects as go
Top_Location_Of_tweet=
df1['place'].value_counts().head (10)

print(Top_Location_Of_tweet)

from nltk. corpus import stopwords
stop = stopwords.words('english')
df1['renderedContent'].apply(lambda x: [item for
item in x if item not in stop])
df1.shape

!pip install tweet-preprocessor

#Remove unnecessary characters
punct = ['%','/',':','\\','&amp','&',';','?']
def remove_punctuations(text):
  for punctuation in punct:
```

```
    text = text.replace(punctuation,'')
  return text
df1['renderedContent'] =
df1['renderedContent'].apply(lambda x:
remove_punctuations(x))
df1['renderedContent'].replace( '', np.nan,
inplace=True)
df1.dropna(subset=["renderedContent"],inplace=True
)
len(df1)

df1 = df1.reset_index(drop=True)
df1.head()

from sklearn.feature_extraction. text import
TfidfVectorizer, CountVectorizer

# Commented out IPython magic to ensure Python
compatibility.
sns.set_style('whitegrid')
# %matplotlib inline
stop=stop+['assured' , 'health' , 'http' , '2023',
'best' ,'look' , 'union', 'product' , 'customer' ,
'india']

def plot_20_most_common_words(count_data,
count_vectorizer) :
```

```
  import matplotlib. pyplot as plt
  words = count_vectorizer.get_feature_names()
  total_counts = np. zeros(len(words))
  for t in count_data:
    total_counts = t.toarray()[0]
  count_dict = (zip(words, total_counts))
  count_dict = sorted(count_dict, key=lambda
x:x[1],reverse=True)[0:20]
  words = [w[0] for w in count_dict]
  counts = [w[1] for w in count_dict]
  x_pos = np.arange(len(words))
  plt.figure(2, (40,40))
  plt.subplot(title = '20 most common words')
  sns. set_context('notebook',font_scale=4,rc={
'lines.linewidth' :2.5})
  sns.barplot(x_pos, counts, palette='husl')
  plt.xticks(x_pos, words, rotation=90)
  plt.xlabel('words')
  plt.ylabel('counts')
  plt.show()
count_vectorizer =
CountVectorizer(stop_words=stop)
# Fit and transform the processed titles
count_data =
count_vectorizer.fit_transform(df1['renderedConten
t'])
# print(count_vectorizer)
```

```
# print(count_data)
# Visualise the 20 most common words
plot_20_most_common_words(count_data,count_vectori
zer)
plt.savefig( 'saved_figure.png')

import cufflinks as cf
cf.go_offline()
cf.set_config_file(offline=False,
world_readable=True)
def get_top_n_bigram(corpus, n=None) :
  vec = CountVectorizer(ngram_range=(2, 4),
stop_words="english").fit(corpus)
  bag_of_words = vec.transform(corpus)
  sum_words = bag_of_words.sum(axis=0)
  words_freq =[(word, sum_words[0, idx]) for word,
idx in vec.vocabulary_.items()]
  words_freq =sorted(words_freq, key = lambda x:
x[1], reverse=True)
  return words_freq[:n]

common_words =
get_top_n_bigram(df1['renderedContent'] , 8)
mydict={}
for word, freq in common_words:
  bigram_df = pd.DataFrame(common_words,columns =
['ngram', 'count'])
```

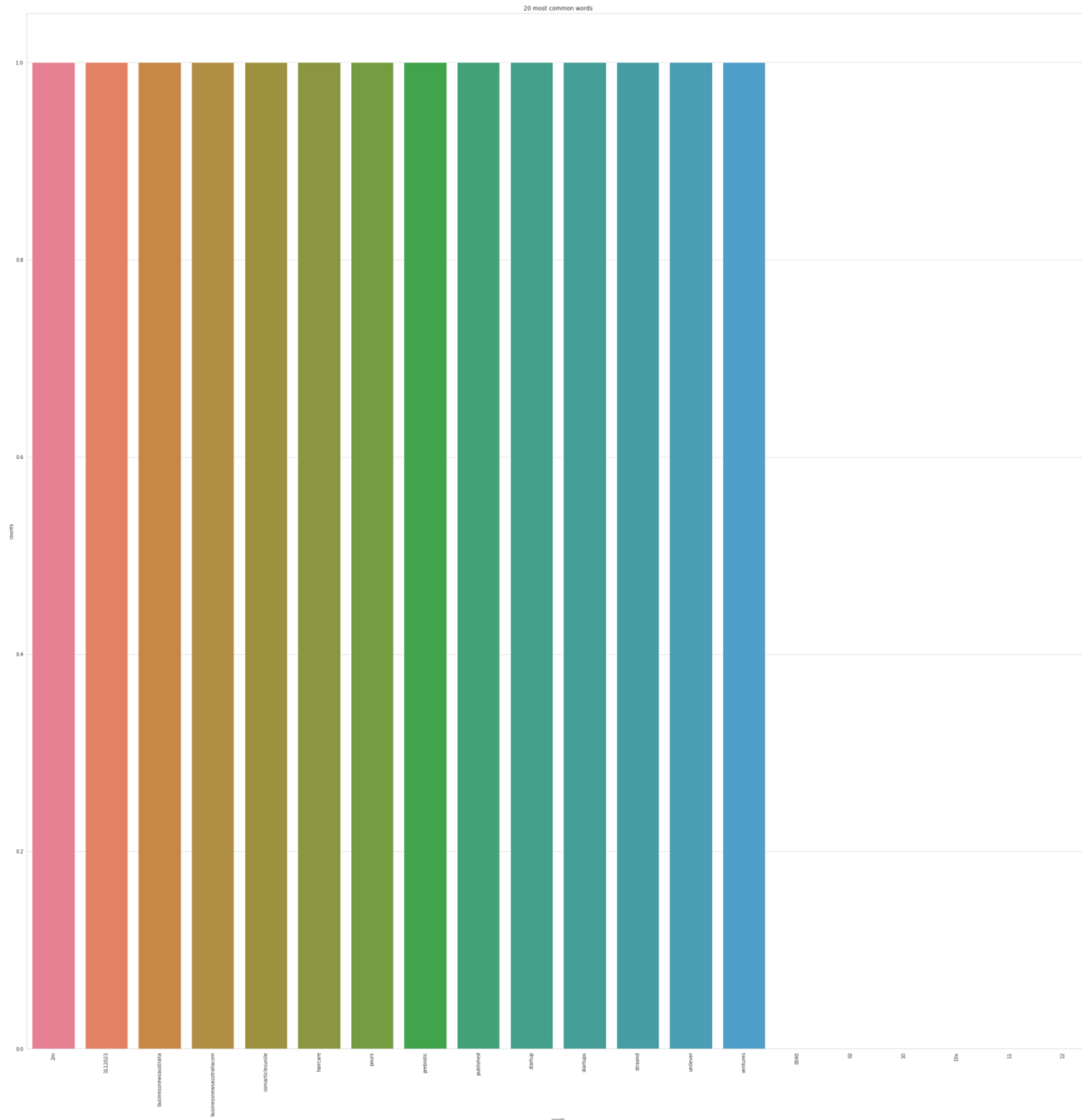
```
bigram_df.groupby( 'ngram'
).sum()['count'].sort_values(ascending=False).sort
_values().plot.barh(title = 'Top 8
bigrams',color='orange')

def get_subjectivity(text):
  return TextBlob(text).sentiment.subjectivity
def get_polarity(text):
  return TextBlob(text).sentiment.polarity
df1['subjectivity']=df1[
'renderedContent'].apply(get_subjectivity)
df1[ 'polarity' ]=df1[
'renderedContent'].apply(get_polarity)
df1.head()

df1['textblob_score'] =df1[
'renderedContent'].apply(lambda x:
TextBlob(x).sentiment.polarity)
neutral_threshold=0.05

df1['textblob_sentiment']=df1[ 'textblob_score']

textblob_df =
df1[['renderedContent','textblob_sentiment','likeC
ount']]

textblob_df["textblob_sentiment"].value_counts()
```

```
df_positive=textblob_df[textblob_df['textblob_sentiment']=='positive' ]
df_very_positive=df_positive[df_positive['likeCount']>0]

df_negative=textblob_df[textblob_df['textblob_sentiment']=='Negative' ]

df_neutral=textblob_df[textblob_df['textblob_sentiment']=='Neutral' ]

#Creating the text variable
positive_tw
="Social,Media,Analytics,Calculation,productivity,financially,physically,emotionally,socially,privacy,Brief,Method,Texts,Analysis"
# Creating word _ cloud with text as argument in .generate() rtpthod
word_cloud1 = WordCloud(collocations = False, background_color = 'white') .generate(positive_tw)
# Display the generated Word Cloud
plt. imshow(word_cloud1, interpolation='bilinear')
plt.axis('off')
plt.show()

#Creating the text variable
```

```
negative_tw
="productivity,financially,physically,emotionally,
socially,privacy,neutrality,algorithms
,Social,Media,Analytics,Calculation, "
# Creating word _ cloud with text as argument in .
generate() rtpthod
word_cloud2 = WordCloud(collocations = False,
background_color = 'white') .generate(negative_tw)
# Display the generated Word Cloud
plt. imshow(word_cloud2, interpolation='bilinear')
plt.axis('off')
plt.show()

#Creating the text variable
neutral_tw ="neutrality,algorithms
,Social,Media,Analytics,Calculation,productivity,f
inancially,physically"
# Creating word _ cloud with text as argument in .
generate() rtpthod
word_cloud2 = WordCloud(collocations = False,
background_color = 'white') .generate(neutral_tw)
# Display the generated Word Cloud
plt. imshow(word_cloud2, interpolation='bilinear')
plt.axis('off')
plt.show()
```

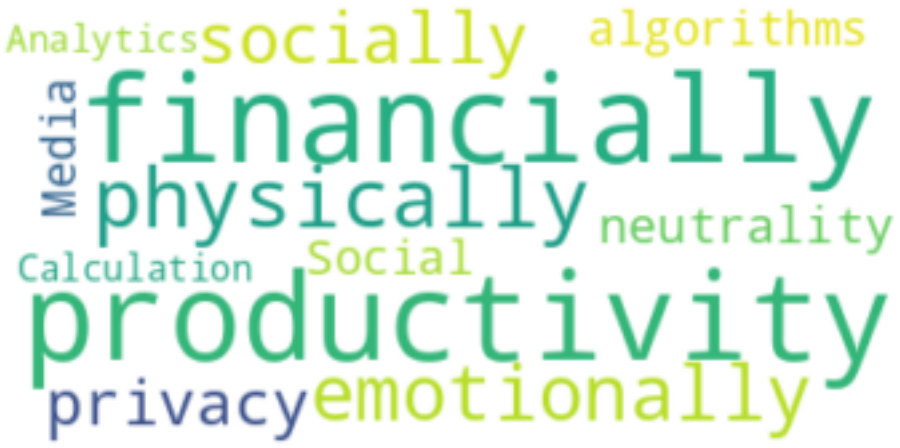
Output:

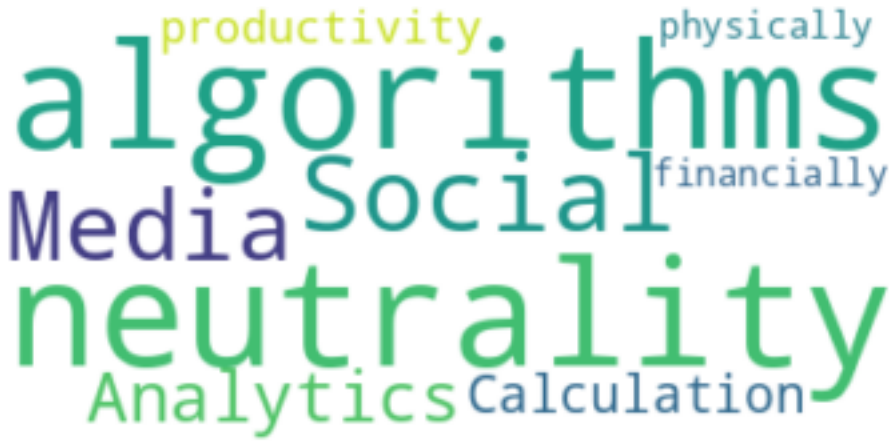

Places of missing values in column
date
rawContent
renderedContent
user
replyCount
retweetCount
likeCount
lang
place
Column_Name
1.0
0.8
0.6
0.4
0.2
0.0


20 most common words
counts
1.0
0.8
0.6
0.4
0.2
0.0
2m
3112023
businessnewsaustralia
businessnewsaustraliacom
comarticlesunile
haircare
pours
prebiotic
published
startup
startups
straand
unilever
ventures
0040
02
10
10x
11
12
words


Top 8 bigrams
ngram
summit 2023
supply chain
hein schumacher
corporate affairs
communications corporate affairs
communications corporate
trends plant
feb 2023
0
5


Analytics
Calculation
productivity
Method
emotionally
socially
privacy
Social
Texts
Media
financially
Brief
physically
Analysis


Analytics
socially
algorithms
financially
Media
physically
neutrality
Calculation
Social
productivity
privacy
emotionally



Conclusion: We have successfully created content based social media analytics model for business.