Experiment No: 3

Aim: Data Cleaning and Storage- Preprocess, filter and store social media data for business (Using Python, MongoDB, R, etc). OR
Exploratory Data Analysis and visualization of Social Media Data for business.

Theory:

- Data cleaning and preprocessing is an essential – and often crucial – part of any analytical process. Social media contains different types of data: information about user profiles, statistics
- (number of likes or number of followers), verbatims, and other media content.
- Quantitative data is very convenient for an analysis using statistical and numerical methods, but unstructured data such as user comments is much more challenging.
- To get meaningful information, one has to perform the whole process of information retrieval. It starts with the definition of the data type and data structure.
- On social media, unstructured data is related to text, images, videos, and sound and wewill mostly deal with textual data.
- Then, the data has to be cleaned and normalized.
- Exploratory Data Analysis (EDA) is usually the first step when you have data in hand and want to analyze it.
- In EDA, there is no hypothesis and no model. You are finding patterns and truth from the data.
- EDA is crucial for data science projects because it can:
- Help you gain intuition about the data;
- Make comparisons between distributions;
- Check if the data is on the scale you expect;
- Find out where data is missing or if there are outliers;
- Summarize data, calculate the mean, min, max, and variance.
- The basic tools of EDA are plots, graphs, and summary statistics.

Preprocessing

- Preprocessing is one of the most important parts of the analysis process.
- It reformats the unstructured data into uniform, standardized form.
- The characters, words, and sentences identified at this stage are the fundamental unitspassed to all further processing stages.
- The quality of the preprocessing has a big impact of the final result on the whole process.
- There are several stages of the process: from simple text cleaning by removing white spaces, punctuation, HTML tags and special characters up to more sophisticated normalization techniques such as tokenization, stemming or lemmatization.

Twitter Data Analytics using Python Steps

- Twitter Data extraction using snscrape Python library
- Twitter Data Cleaning and Preprocessing using Python
- Twitter Data Visualization
- Twitter Data Sentiment Analysis using Textblob

Program Code

1. Scrape Twitter Data for Union Budget 2023

```
!pip install snscrape
import pandas as pd
import snscrape.modules.twitter as sntwitterimport
numpy as np
import matplotlib.pyplot as plt import
seaborn as sns
import nltk nltk.download('stopwords')
from
from nltk.corpus import stopwords from
nltk.tokenize import word_tokenize from nltk.stem
import WordNetLemmatizer from nltk.stem.porter
import PorterStemmerimport string
import re import
textblob
from textblob import TextBlobimport 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'Budget 2023
until:2023-02-07'>text-query-tweets.json")
tweets_df = pd.read_json("text-query-tweets.json" ,lines=True)tweets_df.head(5)
tweets_df.to_csv()
```

2. Data Loading

```
df1 = tweets_df[[ 'date', 'rawContent' , 'renderedContent' , 'user' , 'replyCount'
,'retweetCount' , 'likeCount' , 'lang' , 'place' , 'hashtags' , 'viewCount']].copy()df1.head()
df1.shape
```

3. Twitter Data Cleaning, Preprocessing and Exploratory Data Analysis

```
df1=df1.drop_duplicates("renderedContent")
df1.shape
df1.head
df1.info
df1.date.value_counts() 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)
```

Twitter Data Cleaning and Preprocessing

```
from nltk. corpus import stopwordsstop =
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, CountVectorizersns.set_style('whitegrid')
%matplotlib inline
stop=stop+['budget2023' , 'budget' , 'httpst' , '2023', 'modi' ,'nsitaraman' , 'union','pmindia' ,
'tax' , '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['renderedContent']) #
```

```
print(count_vectorizer)
# print(count_data)
#       Visualise       the       20       most       common       words
plot_20_most_common_words(count_data,count_vectorizer)       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 invec.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 8bigrams',color='orange' ,
width=.4, figsize=(12,8),stacked = True)
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'].apply(lambda    c:'positive'    if    c    >=
neutral_threshold else ('Negative' if c <= -(neutral_threshold) else 'Neutral' ) ) textblob_df =
df1[['renderedContent','textblob_sentiment','likeCount']] textblob_df
textblob_df["textblob_sentiment"].value_counts()
textblob_df["textblob_sentiment"].value_counts().plot.barh(title      =      'Sentiment
Analysis',color='orange'     ,     width=.4,     figsize=(12,8),stacked     =     True)
df_positive=textblob_df[textblob_df['textblob_sentiment']=='positive'                  ]
df_very_positive=df_positive[df_positive['likeCount']>0] df_very_positive.head()
```