Elon Musk Tweet Analysis

In this Project we analyse what makes up Elon Musk Twitter Profile.

```
#installing basic libraries
!pip install nltk
!pip install tweepy
!pip install configparser
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/p
Requirement already satisfied: nltk in /usr/local/lib/python3.7/dist-packages (3.7)
Requirement already satisfied: click in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: joblib in /usr/local/lib/python3.7/dist-packages (fron
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (from r
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.7/dist-packa
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/p
Requirement already satisfied: tweepy in /usr/local/lib/python3.7/dist-packages (3.1(
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.7/c
Requirement already satisfied: requests[socks]>=2.11.1 in /usr/local/lib/python3.7/di
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: PySocks!=1.5.7,>=1.5.6 in /usr/local/lib/python3.7/dis
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/p
Requirement already satisfied: configparser in /usr/local/lib/python3.7/dist-packages
```

- Tweepy library is for accessing and writing tweets from Twitter API.
- configparser for reading config files(which we use to read API keys and keep personal)
- NLTK for standard natural language Processing package

```
#import basic libraries
import tweepy
import configparser
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
#Reading Config file which contains API keys and access token privately
config=configparser.ConfigParser()
config.read("/content/config.ini")
```

```
['/content/config.ini']
```

```
#Accessing APi key and key secret
api_key=config["twitter"]["api_key"]
api_key_secret=config["twitter"]["api_key_secret"]
```

```
#Accessing access_token and token secret
```

```
access_token=config["twitter"]["access_token"]
access_token_secret=config["twitter"]["access_token_secret"]

#Creating an auth instance
auth=tweepy.OAuthHandler(api_key,api_key_secret)
auth.set_access_token(access_token,access_token_secret)


#Creating API instance
api=tweepy.API(auth)


#Here we intialized user with required twitter user details and limit number of tweets to
user="elonmusk"
limit=4000


#tweets are initialized with all the tweets that are accessed using API
tweets=api.user_timeline(screen_name=user, count=limit,tweet_mode="extended")
tweets=tweepy.Cursor(api.user_timeline,screen_name=user, count=3000,tweet_mode="extended")


#we would access 3 elements from the tweets Username,Tweet and Time of tweet
columns=["User","Tweet","Created_time"]
data=[]


#Splitting and storing all the tweets in data list object
for tweet in tweets:
  print(tweet.full_text)
  data.append([tweet.user.screen_name,tweet.full_text,tweet.created_at])
```

```
@cnunezimages @SpaceX @SpaceIntellige3
@Rainmaker1973 Shanghai is beautiful
@PPathole Probably way sooner before it’s too hot for civilization
RT @SpaceX: All systems and weather are looking good ahead of tonight’s launch of
Unless susceptible to extreme natural disasters, nuclear power plants should not b
@WatcherGuru Taxing all billionaires at 100% only drops national debt by ~10%, whi
@WatcherGuru This is scary, something’s got to give
Nothing is more permanent than a “temporary” government program
@traderjourney Exactly!
There is a lot of accounting trickery in this bill that isn’t being disclosed to t
If “temporary” provisions in the Build Back Better Act become permanent, US nation
@engineers_feed Judith Cohen (Jack Black’s mother) also did important work on Apol
@28delayslater https://t.co/jvSALWJFCj
@28delayslater https://t.co/Ej9SWAbcfM
@28delayslater https://t.co/mO4bI8MNqT
@28delayslater
https://t.co/FJaW6L5ba0
@waitbutwhy Brain transplants
@jessica_kirsh @SpaceX Booster production is currently ahead of engine production
@tesla_raj Lot of people don’t realize that you can watch almost any show in a Tes
@WorldAndScience If you leave hydrogen out in the sun long enough, it starts talki
@Rainmaker1973 Wow
@tobyliiiiiiiiii @RGVaerialphotos @SpaceX Hopefully, this month, no later than nex
@RGVaerialphotos @SpaceX Progress
@joshdcaplan It’s true
@BillyM2k
@WholeMarsBlog
```

```
@teslafsdbeta Replacing faulty/missing neurons with circuits is the right way to t

Progress will accelerate when we have devices in humans (hard to have nuanced conv
@newsmax There are already minimum age requirements for the House, Senate &amp; Pr
@stocktalkweekly @neuralink I am definitely not saying that we can for sure do thi
@ICannot_Enough @kimpaquette Exactly
@UniverCurious That always blows my mind. Sad thing is that we haven't been back t
@teslaownersSV Did it myself
@Beniko26020660 10.6.1 coming in a few days to address a few annoying issues
@Rainmaker1973 Looking forward to visiting. I've heard it's awesome.
A background in "AI" is not needed, just exceptional skill in software or computer
@NASA @NASA_Astronauts Congratulations!
@DrSallyL @Tesla Coming soon. Lot of cool stuff.
@kimpaquette Tesla publishes accident statistics quarterly. They are so much bette
As always, Tesla is looking for hardcore AI engineers who care about solving probl
https://t.co/0B5toOOHcj
@AEIecon @SciGuySpace @JimPethokoukis @PE_Podcast_AEI Good summary
@EPavlic He is quite a bossy dog :)
@BillyM2k NFTs are jpeging the dollar
@BillyM2k
@ErcXspace Landing on tower arms
@muratpak My car is currently orbiting Mars
@muratpak You betcha
Starships to ♥ Mars ♥
We will soon make these real https://t.co/t4z5oNFnwW
https://t.co/sIGZPDyx76
@ID_AA_Carmack Haha pretty much
@Kristennetten @MinimalDuck @LudaLisl @28delayslater @JohnnaCrider1 @arctechinc @a
@joroulette It is an honor to serve NASA and the countries of the International Sp
@NASASpaceflight 39A is hallowed spaceflight ground – no place more deserving of a
```

```python
#Creating a dataframe with the given data
df=pd.DataFrame(data,columns=columns)
```

```python
df
```

```
        User                                  Tweet        Created_time
```

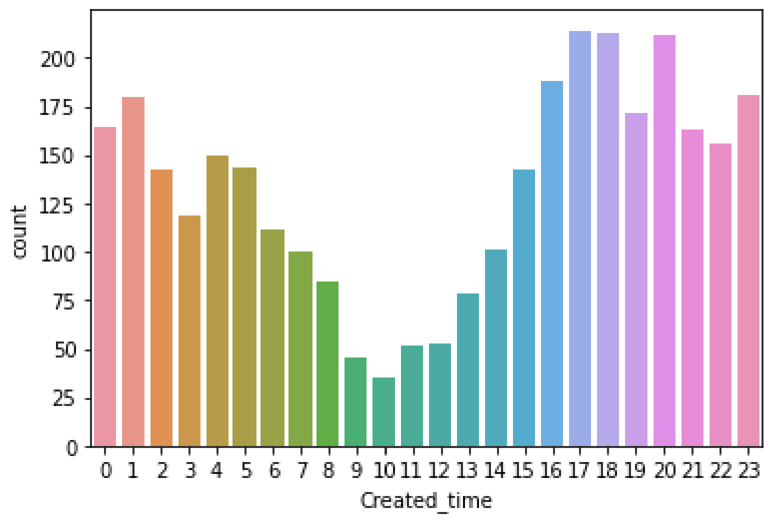
```
print(df.head(10))
```

```
       User                                          Tweet  \
0  elonmusk         @Rainmaker1973 Such an incredible engine!
1  elonmusk                        RT @Tesla: Powerwall FTW!
2  elonmusk                                     @ajtourville
3  elonmusk  RT @Tesla: AI Day tomorrow https://t.co/oVenZD...
4  elonmusk                    @NASAHubble @NASA @SpaceX Yay
5  elonmusk                          @MuskUniversity True
6  elonmusk  Needs be able to get from Starbase to South Pa...
7  elonmusk                    @WholeMarsBlog Off-label use
8  elonmusk  Cybertruck will be waterproof enough to serve ...
9  elonmusk  @phibetakitten Submarines use electric motors ...

         Created_time
0 2022-09-30 15:40:12
1 2022-09-30 14:41:37
2 2022-09-30 05:41:09
3 2022-09-30 05:37:30
4 2022-09-30 01:15:33
5 2022-09-29 20:22:46
6 2022-09-29 15:35:09
7 2022-09-29 15:32:42
8 2022-09-29 15:31:12
9 2022-09-29 15:28:10
```

Analyse the time column:

```
#Analysing Hours in which he is more active
import seaborn as sns
sns.countplot(pd.DatetimeIndex(df["Created_time"]).hour)
```

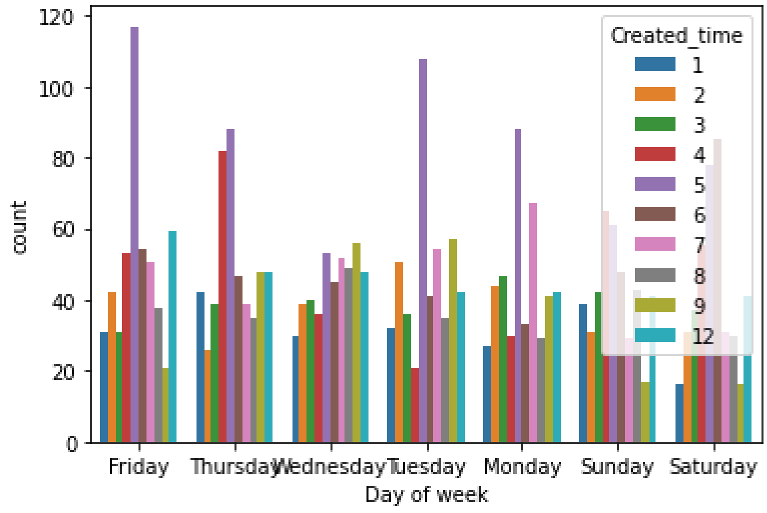
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7ff43c9be510>
```



We can observe that musk is active in twitter from 15:00:00 to 5:00:00 UTC which comes out to be around 9:00:00 to 23:00:00 MDT

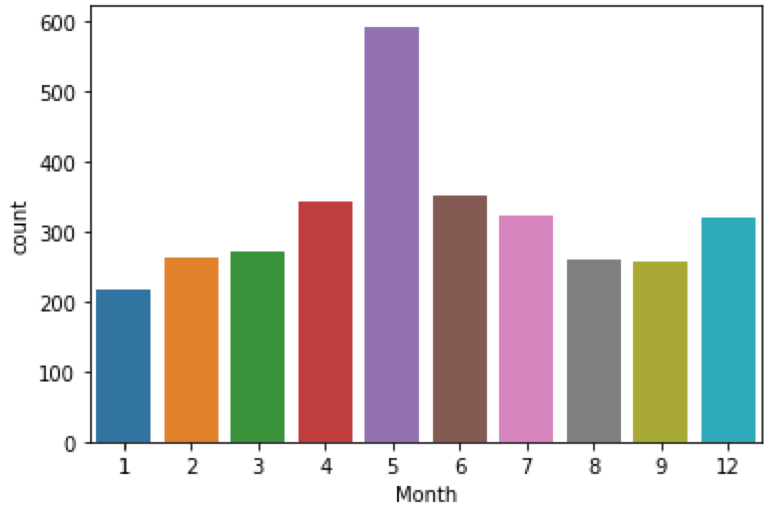
```
#Month to day of week wise Analysis
ax=sns.countplot(pd.DatetimeIndex(df["Created_time"]).day_name(),hue=pd.DatetimeIndex(df["
ax.set_xlabel("Day of week")
```

```
    /usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pas
      FutureWarning
    Text(0.5, 0, 'Day of week')
```



```
#Twitter posts by month
ax=sns.countplot(pd.DatetimeIndex(df["Created_time"]).month)
ax.set_xlabel("Month")
```

```
    /usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pas
      FutureWarning
    Text(0.5, 0, 'Month')
```



He is more active in May 2022 than any other month .

Tweets Analysis

Data processing & cleaning

- Converting html entities
- Removing "@user" from all the tweets
- Changing all the tweets into lowercase
- Apostrophe ,Short Word & Emoticon Lookup
- Replacing Special Characters with space
- Replacing Numbers (integers) with space
- Removing words whom length is less than 2

```
#importing HTMLParser
from html.parser import HTMLParser
html_parser = HTMLParser()
```

```
df["new_tweet"]=df["Tweet"].apply(lambda x: html_parser.unescape(x))
```

```
    /usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: DeprecationWarning: 
      """Entry point for launching an IPython kernel.
```

```
df
```

	User	Tweet	Created_time	new_tweet
0	elonmusk	@Rainmaker1973 Such an incredible engine!	2022-09-30 15:40:12	@Rainmaker1973 Such an incredible engine!
1	elonmusk	RT @Tesla: Powerwall FTW!	2022-09-30 14:41:37	RT @Tesla: Powerwall FTW!
2	elonmusk	@ajtourville	2022-09-30 05:41:09	@ajtourville
3	elonmusk	RT @Tesla: AI Day tomorrow https://t.co/oVenZD...	2022-09-30 05:37:30	RT @Tesla: AI Day tomorrow https://t.co/oVenZD...
4	elonmusk	@NASAHubble @NASA @SpaceX Yay	2022-09-30 01:15:33	@NASAHubble @NASA @SpaceX Yay
...	...	...	...	...
3195	elonmusk	@WholeMarsBlog “Insane technology bandwagon”	2021-12-03 15:56:06	@WholeMarsBlog “Insane technology bandwagon”
3196	elonmusk	@WholeMarsBlog Initial production will be 4 mo...	2021-12-03 15:55:23	@WholeMarsBlog Initial production will be 4 mo...

```
#Function to remove @
```

```
import regex as re
def removepatern(txt_in,patrn):
  r=re.findall(patrn,txt_in)
  for i in r:
    txt_in=re.sub(i," ",txt_in)
  return txt_in
```

```
#Using Regex and vectorize to replace @usernames to blank spaces

df["new_tweet"]=np.vectorize(removepatern)(df["new_tweet"],"@[\w]*")
```

```
df.head(10)
```

	User	Tweet	Created_time	new_tweet
0	elonmusk	@Rainmaker1973 Such an incredible engine!	2022-09-30 15:40:12	Such an incredible engine!
1	elonmusk	RT @Tesla: Powerwall FTW!	2022-09-30 14:41:37	RT : Powerwall FTW!
2	elonmusk	@ajtourville	2022-09-30 05:41:09	
3	elonmusk	RT @Tesla: AI Day tomorrow https://t.co/oVenZD...	2022-09-30 05:37:30	RT : AI Day tomorrow https://t.co/oVenZDbVMQ
4	elonmusk	@NASAHubble @NASA @SpaceX Yay	2022-09-30 01:15:33	Yay
5	elonmusk	@MuskUniversity True	2022-09-29 20:22:46	True
6	elonmusk	Needs be able to get from	2022-09-29	Needs be able to get from

```
#Converting all the tweets to lowercase
df["new_tweet"]=df["new_tweet"].str.lower()
```

```
#Creating a dictionary containing all apostrophes
#Got all the data online
apostrophe_dict = {
"ain't": "am not / are not",
"aren't": "are not / am not",
"can't": "cannot",
"can't've": "cannot have",
"'cause": "because",
"could've": "could have",
"couldn't": "could not",
"couldn't've": "could not have",
"didn't": "did not",
"doesn't": "does not",
"don't": "do not",
"hadn't": "had not",
"hadn't've": "had not have",
"hasn't": "has not",
"haven't": "have not",
"he'd": "he had / he would",
"he'd've": "he would have",
"he'll": "he shall / he will",
"he'll've": "he shall have / he will have",
"he's": "he has / he is",
"how'd": "how did",
"how'd'y": "how do you",
"how'll": "how will",
"how's": "how has / how is",
"i'd": "I had / I would",
"i'd've": "I would have",
"i'll": "I shall / I will",
"i'll've": "I shall have / I will have",
"i'm": "I am",
"i've": "I have",
"isn't": "is not",
"it'd": "it had / it would",
"it'd've": "it would have",
"it'll": "it shall / it will",
"it'll've": "it shall have / it will have",
"it's": "it has / it is",
"let's": "let us",
"ma'am": "madam",
"mayn't": "may not",
"might've": "might have",
"mightn't": "might not",
"mightn't've": "might not have",
"must've": "must have",
"mustn't": "must not",
"mustn't've": "must not have",
"needn't": "need not",
"needn't've": "need not have",
"o'clock": "of the clock",
"oughtn't": "ought not",
"oughtn't've": "ought not have",
"shan't": "shall not",
"sha'n't": "shall not",
"shan't've": "shall not have",
"she'd": "she had / she would",
"she'd've": "she would have",
"she'll": "she shall / she will",
"she'll've": "she shall have / she will have",
"she's": "she has / she is",
"should've": "should have",
"shouldn't": "should not",
"shouldn't've": "should not have",
"so've": "so have",
"so's": "so as / so is",
"that'd": "that would / that had",
"that'd've": "that would have",
"that's": "that has / that is",
"there'd": "there had / there would",
"there'd've": "there would have",
"there's": "there has / there is",
"they'd": "they had / they would",
"they'd've": "they would have",
"they'll": "they shall / they will",
"they'll've": "they shall have / they will have",
"they're": "they are",
"they've": "they have",
"to've": "to have",
"wasn't": "was not",
"we'd": "we had / we would",
"we'd've": "we would have",
"we'll": "we will",
"we'll've": "we will have",
"we're": "we are",
"we've": "we have",
"weren't": "were not",
"what'll": "what shall / what will",
"what'll've": "what shall have / what will have",
"what're": "what are",
"what's": "what has / what is",
"what've": "what have",
"when's": "when has / when is",
"when've": "when have",
"where'd": "where did",
"where's": "where has / where is",
"where've": "where have",
"who'll": "who shall / who will",
"who'll've": "who shall have / who will have",
"who's": "who has / who is",
"who've": "who have",
"why's": "why has / why is",
"why've": "why have",
"will've": "will have",
"won't": "will not",
"won't've": "will not have",
"would've": "would have",
"wouldn't": "would not",
"wouldn't've": "would not have",
"y'all": "you all",
"y'all'd": "you all would",
"y'all'd've": "you all would have",
"y'all're": "you all are",
"y'all've": "you all have",
"you'd": "you had / you would",
"you'd've": "you would have",
"you'll": "you shall / you will",
"you'll've": "you shall have / you will have",
"you're": "you are",
"you've": "you have"}

#Function to convert apostrophe to with apostrophe
def lookup_dict(text, dictionary):
    for word in text.split():
```

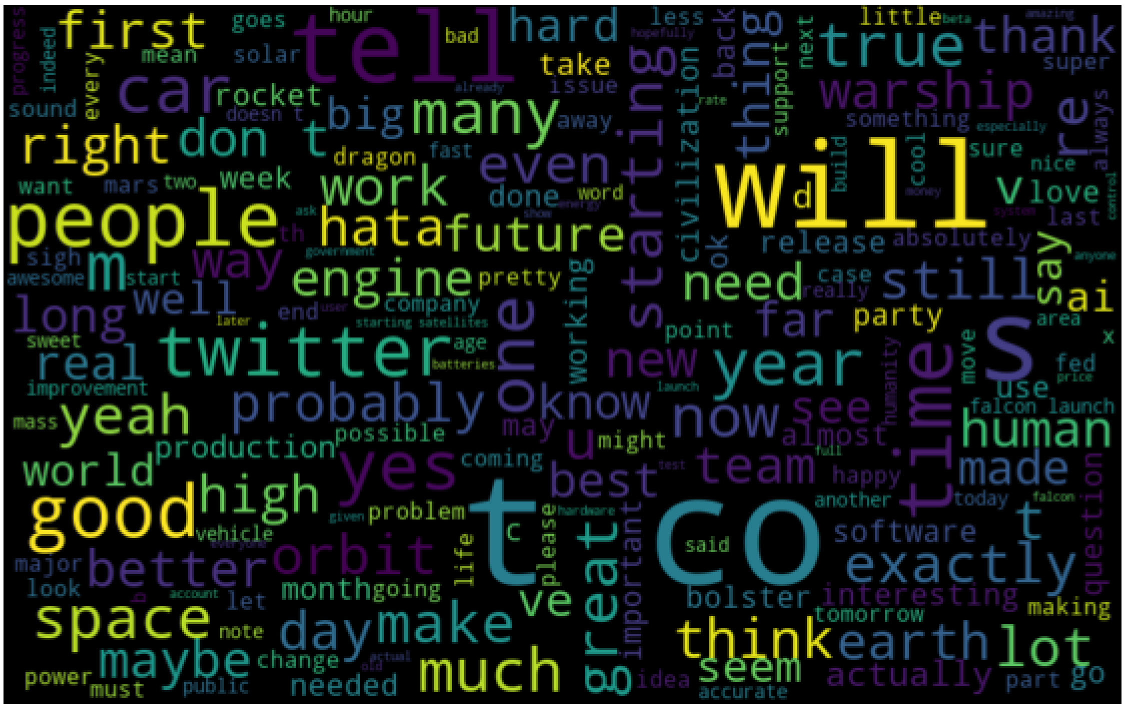
```
        if word.lower() in dictionary:
            if word.lower() in text.split():
                text = text.replace(word, dictionary[word.lower()])
    return text

#Converting Aphostrophe
df["new_tweet"]=df["new_tweet"].apply(lambda x: lookup_dict(x,apostrophe_dict))


# Creating Short word Dictonary
short_word_dict = {
"121": "one to one",
"a/s/l": "age, sex, location",
"adn": "any day now",
"afaik": "as far as I know",
"afk": "away from keyboard",
"aight": "alright",
"alol": "actually laughing out loud",
"b4": "before",
"b4n": "bye for now",
"bak": "back at the keyboard",
"bf": "boyfriend",
"bff": "best friends forever",
"bfn": "bye for now",
"bg": "big grin",
"bta": "but then again",
"btw": "by the way",
"cid": "crying in disgrace",
"cnp": "continued in my next post",
"cp": "chat post",
"cu": "see you",
"cul": "see you later",
"cul8r": "see you later",
"cya": "bye",
"cyo": "see you online",
"dbau": "doing business as usual",
"fud": "fear, uncertainty, and doubt",
"fwiw": "for what it's worth",
"fyi": "for your information",
"g": "grin",
"g2g": "got to go",
"ga": "go ahead",
"gal": "get a life",
"gf": "girlfriend",
"gfn": "gone for now",
"gmbo": "giggling my butt off",
"gmta": "great minds think alike",
"h8": "hate",
"hagn": "have a good night",
"hdop": "help delete online predators",
"hhis": "hanging head in shame",
"iac": "in any case",
"ianal": "I am not a lawyer",
"ic": "I see",
"idk": "I don't know",
"imao": "in my arrogant opinion",
"imnsho": "in my not so humble opinion",
"imo": "in my opinion",
"iow": "in other words",
"ipn": "I’m posting naked",
"irl": "in real life",
"jk": "just kidding",
"l8r": "later",
"ld": "later, dude",
"ldr": "long distance relationship",
"llta": "lots and lots of thunderous applause",
"lmao": "laugh my ass off",
"lmirl": "let's meet in real life",
"lol": "laugh out loud",
"ltr": "longterm relationship",
"lulab": "love you like a brother",
"lulas": "love you like a sister",
"luv": "love",
"m/f": "male or female",
"m8": "mate",
"milf": "mother I would like to fuck",
"oll": "online love",
"omg": "oh my god",
"otoh": "on the other hand",
"pir": "parent in room",
"ppl": "people",
"r": "are",
"rofl": "roll on the floor laughing",
"rpg": "role playing games",
"ru": "are you",
"shid": "slaps head in disgust",
"somy": "sick of me yet",
"sot": "short of time",
"thanx": "thanks",
"thx": "thanks",
"ttyl": "talk to you later",
"u": "you",
"ur": "you are",
"uw": "you’re welcome",
"wb": "welcome back",
"wfm": "works for me",
"wibni": "wouldn't it be nice if",
"wtf": "what the fuck",
"wtg": "way to go",
"wtgp": "want to go private",
"ym": "young man",
"gr8": "great"
}


#Converting short words
df["new_tweet"]=df["new_tweet"].apply(lambda x: lookup_dict(x,short_word_dict))


#Creating Emoticon Dict
```

```
emoticon_dict = {
":)": "happy",
":-)": "happy",
":-]": "happy",
":-3": "happy",
":->": "happy",
"8-)": "happy",
":-}": "happy",
":o)": "happy",
":c)": "happy",
":^)": "happy",
"=]": "happy",
"=)": "happy",
"<3": "happy",
":-(": "sad",
":(": "sad",
":c": "sad",
":<": "sad",
":[": "sad",
">:[": "sad",
":{": "sad",
">:(": "sad",
":-c": "sad",
":-< ": "sad",
":-[": "sad",
":-||": "sad"
}


#Find and replace emoticon dict
df["new_tweet"]=df["new_tweet"].apply(lambda x: lookup_dict(x,emoticon_dict))


#Finding and replacing all symbols,numbers and integers with " "
df["new_tweet"]=df["new_tweet"].apply(lambda x: re.sub(r'[^\w\s]',' ',x))
df["new_tweet"]=df["new_tweet"].apply(lambda x: re.sub(r'[^a-zA-Z0-9]',' ',x))
df["new_tweet"]=df["new_tweet"].apply(lambda x: re.sub(r'[^a-zA-Z]',' ',x))


#Dropping words less than 2
df["new_tweet"]=df["new_tweet"].apply(lambda x: " ".join([i for i in x.split() if len(x)>=


#importing textblob
from textblob import TextBlob



#Correcting all words with corrected words
df["new_tweet"]=df["new_tweet"].apply(lambda x: str(TextBlob(x).correct()))


#After processing and cleaning DataFRame becomes
df
```

	User	Tweet	Created_time	new_tweet
0	elonmusk	@Rainmaker1973 Such an incredible engine!	2022-09-30 15:40:12	such an incredible engine
1	elonmusk	RT @Tesla: Powerwall FTW!	2022-09-30 14:41:37	it powerwall few
2	elonmusk	@ajtourville	2022-09-30 05:41:09	
3	elonmusk	RT @Tesla: AI Day tomorrow https://t.co/oVenZD...	2022-09-30 05:37:30	it ai day tomorrow http t co ovenzdbvmq
4	elonmusk	@NASAHubble @NASA @SpaceX Yay	2022-09-30 01:15:33	may
...	...	...	...	...
3195	elonmusk	@WholeMarsBlog “Insane technology bandwagon”	2021-12-03 15:56:06	insane technology bandwagon
3196	elonmusk	@WholeMarsBlog Initial production will be 4 mo	2021-12-03 15:55:23	initial production will be motor variant with

```
#Graphical Representation to show weightage of each word used
all_words = ' '.join([text for text in df['new_tweet']])
from wordcloud import WordCloud
wordcloud = WordCloud(width=800, height=500, random_state=21, max_font_size=110).generate(

plt.figure(figsize=(10, 7))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
plt.title("Most Common words in column Tweet")
plt.show()
```

Most Common words in column Tweet



```
#importing ntlk library
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import nltk
nltk.download('punkt')
nltk.download('stopwords')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
True
```

```
#Tokenizing words
df["new_tweet"]=df["new_tweet"].apply(lambda x: word_tokenize(x))
```

```
#importing all stopwords
stop_words = set(stopwords.words('english'))
stop_words
```

```
 'shouldn',
 "shouldn't",
 'so',
 'some',
 'such',
 't',
 'than',
 'that',
 "that'll",
 'the',
 'their',
 'theirs',
 'them',
 'themselves',
 'then',
 'there',
 'these',
 'they',
 'this',
 'those',
 'through',
 'to',
 'too',
 'under',
 'until',
 'up',
 've',
 'very',
 'was',
 'wasn',
 "wasn't",
 'we',
 'were',
 'weren',
 "weren't",
 'what',
 'when',
 'where',
 'which',
 'while',
 'who',
 'whom',
 'why',
 'will',
 'with',
 'won',
 "won't",
 'wouldn',
 "wouldn't",
 'y',
 'you',
 "you'd",
 "you'll",
 "you're",
 "you've",
 'your',
 'yours',
 'yourself',
```

```
#Dropping all stop_words
df["new_tweet"]=df["new_tweet"].apply(lambda x: [i for i in x if i not in stop_words])
```

```
#importing Stemmer to convert words to base word
from nltk.stem import PorterStemmer
stemming = PorterStemmer()
```

```
#Creating a column with base word
df["stemmed"]=df["new_tweet"].apply(lambda x: ' '.join([stemming.stem(i) for i in x]))
```

```
#importing WordNetLemmatizer
from nltk.stem.wordnet import WordNetLemmatizer
lemmatizing = WordNetLemmatizer()
```

```
nltk.download('wordnet')
nltk.download('omw-1.4')
```

```
     [nltk_data] Downloading package wordnet to /root/nltk_data...
     [nltk_data] Downloading package omw-1.4 to /root/nltk_data...
     True
```

```
#Creating a column after lemmatization (converting all words to base)
df["lemmatized"]=df["new_tweet"].apply(lambda x: ' '.join([lemmatizing.lemmatize(i) for i
```

```
#Updated  DataFRame
df
```

	User	Tweet	Created_time	new_tweet	stemmed	lemmatized
0	elonmusk	@Rainmaker1973 Such an incredible engine!	2022-09-30 15:40:12	[incredible, engine]	incred engin	incredible engine
1	elonmusk	RT @Tesla: Powerwall FTW!	2022-09-30 14:41:37	[powerwall]	powerwal	powerwall
2	elonmusk	@ajtourville	2022-09-30 05:41:09	[]		
3	elonmusk	RT @Tesla: AI Day tomorrow https://t.co/oVenZD...	2022-09-30 05:37:30	[ai, day, tomorrow, http, co, ovenzdbvmq]	ai day tomorrow http co ovenzdbvmq	ai day tomorrow http co ovenzdbvmq
4	elonmusk	@NASAHubble @NASA @SpaceX Yay	2022-09-30 01:15:33	[may]	may	may
...	...	...	...	...	...	...
3195	elonmusk	@WholeMarsBlog “Insane technology bandwagon”	2021-12-03 15:56:06	[insane, technology, bandwagon]	insan technolog bandwagon	insane technology bandwagon
3196	elonmusk	@WholeMarsBlog Initial production will be 4 mo...	2021-12-03 15:55:23	[initial, production, motor, variant, independ...	initi product motor variant independ ultra fas...	initial production motor variant independent u...
		your po...		happy, see]	see	happy see

Representation text in wordcloud based on weightage of Stemmed Column Text

```
all_words = ' '.join([text for text in df['stemmed']])
from wordcloud import WordCloud
wordcloud = WordCloud(width=800, height=500, random_state=21, max_font_size=110).generate(

plt.figure(figsize=(10, 7))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
plt.title("Most Common words in column Tweet Stemmed")
plt.show()
```

Most Common words in column Tweet Stemmed

Representation text in wordcloud based on weightage of lemmatized Column Text

```
all_words = ' '.join([text for text in df['lemmatized']])
from wordcloud import WordCloud
wordcloud = WordCloud(width=800, height=500, random_state=21, max_font_size=110).generate(

plt.figure(figsize=(10, 7))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
plt.title("Most Common words in column Tweet lemmatized")
plt.show()
```

Most Common words in column Tweet lemmatized